Energy-Efficient Topology Construction via Power Allocation for Decentralized Learning via Smart Devices with Edge Computing

Jian-Jhih Kuo, *Member, IEEE,* Cheng-Wei Ching, *Member, IEEE,* Hung-Sheng Huang, *Graduate Student Member, IEEE,* and Yu-Chun Liu

***Abstract*—Data privacy preservation has drawn much attention in emerging machine learning applications. Decentralized learning among smart devices over wireless networks is thus developed to guarantee data security and eliminate the involvement of parameter servers to avoid transmission bottlenecks. However, the previous research focuses on data compression and exchange rules of model parameters among smart devices. Still, it neglects the interplay between link cardinality, transmission power consumption, and collision in transmission. To jointly optimize these issues, in this paper, we first set collision and interference aside and formulate a new optimization problem, named GreenDL, and then extend GreenDL to be collision-aware, namely, GreenDL-CA, by restricting the maximum degree of each smart devices. We prove their hardness and propose two approximation algorithms dubbed as CoTRAIN and CoTRAIN-CA for GreenDL and GreenDL-CA, respectively. Experiment and simulation results manifest that both CoTRAIN and CoTRAIN-CA reduce more than 20% power compared with the other heuristics without sacrificing the convergence rate in the decentralized learning practices.**

***Index Terms*—Decentralized learning, topology construction, energy conservation, wireless communication, NP-hardness, approximation algorithm**

I. Introduction

Nowadays, artificial intelligence (AI) has drawn much attention and innovates numerous advanced applications [2]. However, AI models usually require considerable amount of data for training, thereby giving rise to two following issues. One is data security and privacy. The data required for training are usually stored in smart devices of users [3]. The leakage of data associated with personal information is thus the last thing that users would like to encounter when enjoying AI-related services [3]. Another is the need of powerful platforms for training. To collect and process enormous amount of data from smart devices *in a centralized fashion*, powerful servers meeting requirements of storage, computing, and bandwidth for emerging service providers are getting more and more impractical when the number of smart devices grows drastically.[1] To ease the above issues, collaborative learning has been proposed to train a target model by multiple smart devices of users with their local data [2], [5], [6]. Usually, a centralized parameter server is required to aggregate different local updates of model parameters from the smart devices for the next round of training, while it may become a crucial network bottleneck and limit the scalability [7].

To this end, decentralized learning (DL) does not aggregate the model in the central server. Each smart device *only* shares local updates of model parameters with its neighboring smart devices in the mobile edge network via device-to-device (D2D) transmission to locally derive *new average model parameters* for the next round of training in DL. Remark that messages here are forwarded to neighboring devices by *one-hop broadcasting* to improve the transmission efficiency and reduce the number of transmissions.[2] It can be envisaged that smart devices act as both a central parameter server and a training unit at the same time. Eventually, DL will converge and achieve *consensus* among the smart devices. DL has two main advantages as follows: 1) It guarantees data privacy since data are only accessed by their owners. 2) The central server for parameter aggregation is no longer required. The following two applications exhibit the advantages of DL concisely. First, DL is exploited to improve cache performance of edge computing for vehicular networks. The nearby vehicles own the private data and collaboratively train a global model with other vehicles via vehicle-to-vehicle (V2V) communications to improve overall content caching scheme [10]. Besides, for human activity recognition, WiFi sensing, and physiological stress detection, employing DL with smart devices via D2D communications mitigates the users' privacy concerns [11], while the edge server built in the base station is required to collect the location information of smart devices and assign the suitable neighbors to each smart device. Afterward, the base station is only responsible for scheduling D2D communications among devices [12] until the target accuracy is achieved.

J.-J. Kuo is with the Department of Computer Science and Information Engineering and the Advanced Institute of Manufacturing with High-tech Innovations, National Chung Cheng University, Taiwan. E-mail: lajacky@cs.ccu.edu.tw.

C.-W. Ching is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. E-mail: g08410092@ccu.edu.tw.

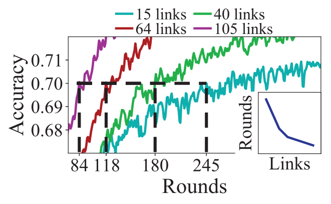
H.-S. Huang is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. E-mail: g08410048@ccu.edu.tw.

Y.-C. Liu is with the Department of Electrical Engineering, National Chung Cheng University, Taiwan. E-mail: u06415067@ccu.edu.tw.

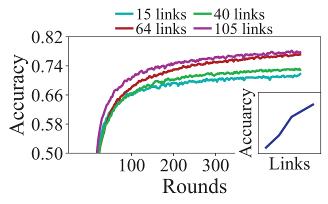
This is an extended version of the conference paper, "Energy-Efficient Link Selection for Decentralized Learning via Smart Devices with Edge Computing" [1] published at the IEEE Global Communications Conference 2020 (GLOBECOM 2020).

[1]Cisco predicts smart devices will grow to 13.1 billions by 2023 [4].

[2]The local update messages can be further compressed for better transmission efficiency [8], [9] while it is beyond the scope of this paper.



(a) Accuracy vs # links



(b) Converged accuracy vs # links

Fig. 1. Effect of number of links ranging $[15, 105]$ on average accuracy in a 16-device network. (a) Assume the target accuracy is 70%, a fewer training rounds comes from a higher links, and (b) a higher converged accuracy comes from a higher links when the prespecified target round is 500.

Intuitively, a topology with more links has more data exchanges and is more likely to yield better training performance (e.g., higher accuracy and fewer training rounds for convergence) for DL. To further verify the interplay between *link cardinality* (i.e., number of links) and training performance, we implemented a DL framework [5] on a small network that consists of 16 smart devices, where the trained model has 2 convolutional layers and 3 fully connected layers and the adopted dataset is CIFAR10 [13]. The more detailed settings about the hyper-parameters can be found in Section V. We evaluate the performance effect on accuracy and loss with different numbers of links (i.e., 15, 40, 64, and 105 links). The effects of different link cardinality on the accuracy and loss are shown in Fig. 1. The accuracy rates of 105-link, 64-link, 40-link, and 15-link topologies achieve 70% at the 84th, 118th, 180th, and 245th rounds, respectively, and eventually converge to 78.1%, 77.8%, 73.1%, and 71.5%, respectively. The above results show that more links in a topology benefit the training performance of DL. To increase the link cardinality in a topology, smart devices are inevitable to use higher transmission power to expand the transmission range to cover more neighboring devices and then incur higher energy consumption during a round, while energy-efficient communications are crucial for smart devices [14]. However, the trade-off between *training performance* and *transmission power consumption* has not been carefully explored for DL to select adequate and energy-efficient links in topology.

Optimizing *transmission power consumption* while ensuring *link cardinality* in the topology for DL to guarantee *training performance* leads to new challenges as follows: 1) *Discrete power levels*. Smart devices have to set a higher transmission power to cover farther smart devices. Nonetheless, smart devices are usually unable to support configurations over a continuous domain but have a *limited*, *discrete* set of possible transmission power levels [15]. A smart device may seriously waste transmission power if covering *inappropriate* smart devices whose locations are slightly beyond the coverage of its previous transmission power level. 2) *Symmetric wireless links*. Raising the transmission power level of only one smart device may not connect one another. Two smart devices have to set a sufficient power level to cover each other in the transmission range. In addition, the topology with the selected links should be *connected*. 3) *Density-aware power selection*. Connecting smart devices *close* to each other tends to reduce the overall transmission power. Moreover, the smart devices in denser areas using high transmission power levels are inclined to have more links than those in sparser areas. Overall, the problem is quite challenging and different from traditional energy-efficient broadcast tree problems[3] since it has to jointly decide which links should be selected to achieve the required *link cardinality* (i.e., not just constructing a spanning tree) to ensure the training performance and which transmission power level should be used for each smart device to minimize the *transmission power consumption*.

To address the challenges, we present **Green** Transmission Power Level Allocation Problem for **D**ecentralized **L**earning (GreenDL). With the given parameters: 1) the smart devices with discrete power levels in the network, 2) the required transmission power level to connect each smart device pair, and 3) a *link cardinality ratio* (i.e., a user-defined ratio of requested link cardinality to total number of possible device pairs), GreenDL asks for a set of links that satisfies the *link cardinality ratio* and yields the minimum overall *transmission power consumption* of smart devices. In addition, collisions and interference may occur frequently when smart devices send the packets (e.g., to exchange model updates with its neighbors in DL) with a large transmission range. This issue may prolong the communication time since transmissions that interfere each other cannot be scheduled *at the same time slot*. Therefore, we further extend GreenDL to consider the issue, namely, GreenDL with collision awareness (GreenDL-CA). Particularly, one more parameter, *the maximum number of nearby smart devices within the transmission range of each smart device*, is required to mitigate the interference during a training round beside the three parameters aforementioned.

Then, we prove that GreenDL and GreenDL-CA are both NP-hard and first design an efficient **Co**llaborative Densi**t**y-Awa**r**e Power Level **A**llocation and L**i**nk Selectio**n** Approximation Algorithm (CoTRAIN) for GreenDL. To jointly address the above three challenges, CoTRAIN introduces a novel notion, *niche link set*, that is, a set of links which connect the smart devices that are close to each other in a *dense* area such that the links have the *lowest power consumption per link cardinality* in the network. Then, CoTRAIN introduces *niche indicator* (detailed later) to evaluate sets of links and recognize the niche link set in the network. Therefore, CoTRAIN iteratively augments the set of selected links with the niche link set until the link cardinality ratio is satisfied. We prove that CoTRAIN achieves a logarithmic approximation ratio (see Section IV). Following the rationale to design CoTRAIN, we introduce CoTRAIN with collision awareness (CoTRAIN-CA) for GreenDL-CA. The proved approximation ratio is also applied to CoTRAIN-CA even though the devices are limited to have at most the prespecified number of neighbors.

On the implementation side, the manifestation of effectiveness of CoTRAIN and CoTRAIN-CA is threefold (see Section V). First, the real-world dataset of Santander City [16] is adopted to evaluate the performance of CoTRAIN/CoTRAIN-CA and other naive heuristics so as to consider the physical

[3]The more detailed discussion and justification are presented in Section III and Appendix A of the supplementary material.

locations of smart devices. Secondly, extensive experiments are conducted to examine the performance of different methods with varying given parameters. Lastly, a lightweight deep neural network (DNN) model is implemented in DL fashion with the sets of links selected by different methods using well-known dataset CIFAR-10 [13]. In addition, to consider the impact of the non-independent-and-identically-distributed (non-IID) data, we follow [17] to model the degree of *data skewness* with which the different methods are evaluated. Overall, CoTRAIN/CoTRAIN-CA are shown to conserve at least 20% of power consumption than do the others while *no* training performance (e.g., accuracy and loss with respect to the classification task) and convergence rate are compromised.

The rest of this paper is organized as follows: Section II exhaustively studies the background and the state of the art in different aspects. The problem formulations and hardness of GreenDL and GreenDL-CA are defined and proved in Section III. Section IV presents the algorithms CoTRAIN and CoTRAIN-CA for GreenDL and GreenDL-CA and analyzes the approximation ratio. The results of simulations are shown in Section V. Finally, Section VI discusses the future work and concludes the paper.

II. Related Work

A. Communication-Efficient Federated Learning Frameworks and Mechanism Designs

Federated Learning (FL) is proposed to train the global model locally via user devices with their data to avoid direct data access [18]. To derive the global model for the next-round learning, FL requires a *parameter server* to coordinate and aggregate the local model updates from user devices. Meng *et al.* show that FL might slow down convergence due to local shuffling of SGD [19]. Many research works focus on different perspectives to optimize the training efficiency. Konečný *et al.* present a method to optimize the transmission efficiency between central servers and devices utilizing lightening transmitted data sizes [20]. Wang *et al.* analyze the convergence rate via distributed gradient descent to achieve the trade-off between the local training epoch and global aggregation [21], [22]. Tran *et al.* decompose FL optimization problems into convex-structured sub-problems [23]. Nishio *et al.* maximize the number of selected devices for a round in FL [24]. However, FL still struggles over the communication bottleneck when the local model updates from the user devices are sent to the parameter server for aggregation. Also, the indispensable requirement of *on-demand* parameter servers is another concern.

B. Communication-Efficient Decentralized Learning Frameworks and Mechanism Designs

DL is first innovated by Tsitsiklis *et al.* [25], and it is also called gossip algorithm. Different from FL, DL does not require a parameter server to aggregate the local model updates from user devices. Nedic *et al.* present an algorithm to quantize (e.g., from 32-bit float point to 8-bit integer) the local update of model parameters for data exchange. Tang *et al.* devise a framework to quantize the difference between the last and current local model parameters to reduce more data exchange [26]. Li *et al.* develop a pipelined framework which allows two consecutive computing iterations to overlap on the timeline and mask the faster of the computation and communication to reduce the training time [6]. Koloskova *et al.* propose a decentralized stochastic gradient descent method, CHOCO-SGD, for quantized and sparsified model updates and theoretically and empirically prove the performance of CHOCO-SGD for convex learning tasks [5]. Koloskova *et al.* further show that CHOCO-SGD achieves a linear speedup of convergence rate in the number of training devices compared to SGD on a single node for arbitrary high compression ratios on general non-convex functions, and non-IID training data [27]. Koloskova *et al.* offer an unified DL framework package for convex and non-convex learning tasks, and the theoretical bound is proved under weak assumptions on gradient descents [28]. However, none of them considers the link selection for data exchange among devices in DL to guarantee the training performance while minimizing the transmission power consumption.

C. Mobile Edge Computing

Mobile Edge Computing (MEC) has emerged to handle the massive data analysis and reduce latency between cloud and user devices [29]–[31]. Xu *et al.* dynamically cache services on the resource-limited edge servers to relieve the overhead in backhaul networks [32]. He *et al.* offload the tasks to the neighboring devices and edge servers to accelerate the completion time [12]. Xiao *et al.* propose a distributed method for fog nodes to jointly optimize energy consumption and user experience [33]. Both S. Wang *et al.* and L. Wang *et al.* aim at the problems of service migration according to user location and mobility for the better user experience [34], [35]. Zhang *et al.* design a decentralized algorithm to orchestrate the resources of multiple different service providers in MEC networks to improve the user experience and maximize total profit for the service providers [36]. However, none of them employs the MEC servers to determine the D2D transmissions among devices for data exchanging in DL so as to offload the data traffic originally relayed by base stations and save more transmission power.

D. Device-to-Device Communications and Applications

Device-to-device (D2D) communication has emerged as an advanced technology since the number of users and applications are growing drastically resulting to the shortage of resources and an increase in power consumption for transmissions [37]. In the field of vehicular ad hoc networks (VANETs), Sun *et al.* propose a D2D-based scheme for power control and resource allocation among vehicle-to-vehicle communication in consideration of the high mobility of vehicles [38]. Sun *et al.* follow the settings in [38] and formulate an optimization problem to maximize sum rate with proportional bandwidth fairness under the constraint of satisfying the vehicular devices' requirements on latency and reliability [39]. Gu *et al.* explore more general use cases in VANETs, and the proposed D2D-based vehicle-to-everything framework considers multiple simultaneous connections for each vehicle equipped with multiple radio interfaces [40]. For Internet of Things, Bello *et al.* exhaustively analyze state-of-the-art D2D communication mechanisms in IoT devices and conclude that they can significantly improve the application robustness and network connectivity [41]. Wang *et al.* propose opportunistic routing algorithms in D2D networks to enable the data traffic of innovative applications, such as video steaming, multiplayer gaming, and so on, to be offloaded from cellular network [42]. Yang *et al.* make use of MEC servers that maintains the information of devices in proximity to tackle the issue of D2D discovery so as to offload the live streaming traffic from mobile networks to D2D networks among mobile devices and improve users' QoE [43]. However, none of them puts focus on the issues of training performance and power efficiency derived from DL over D2D communications.

III. The Scenario, GreenDL Problem, and Its Extension

A. The Scenario

For each iteration in DL, every involved smart device has to share its local updates of model parameters with the assigned devices and aggregate the received updates as the new model for the next-round training. However, in real-world scenarios, smart devices tend to be selfish and have a lower willingness to consume their bandwidth to relay the packets to the others since they are usually equipped with limited resources [44], [45] (e.g., low wireless bandwidth and limited discrete power levels). Moreover, low-cost smart devices are inclined to lose packets frequently and become the traffic bottlenecks of multi-hop routing [46]–[50]. Such traffic bottlenecks can further prolong the synchronous procedure of DL.[4] Therefore, in this paper, smart devices are assumed to share their data (i.e., local model update of parameters) only with the assigned neighboring devices via one-hop D2D transmission instead of multi-hop routing. On the other hand, the edge server built in the base station [21] has to collect the location information of devices and construct an energy-efficient communication topology to indicate the neighboring devices for each device [28]. After that, the base station is supposed to schedule the D2D communications among the devices until the target accuracy is achieved [23].

It is worth noting that unlike the Min-Power Symmetric Connectivity problem (MPSC) [51] (i.e., the traditional minimum-energy broadcast problem, see Definition 1 in Section III-D), The devices in DL require a *connected topology with a set of adequate links* instead of a *spanning tree* alone. Therefore, our problem is more intractable than MPSC because the number of its required links is independent of the number of devices n while that is always $n-1$ (i.e., the number of links in a tree) in MPSC [51]. Moreover, MPSC does not support limited, discrete set of available power levels, which is typical for low-cost smart devices. It thereby motivates us to investigate a novel problem to construct such a communication topology for data exchange among devices to ensure the training performance while minimizing the power consumption.

[4]The detailed discussion and justification for GreenDL and the synchronous procedure of DL are provided in Appendix A of the supplementary material.

B. The Problem Formulation of GreenDL

In the following, we formally define the optimization problem, **Green** Transmission Power Level Allocation Problem for **D**ecentralized **L**earning (GreenDL). GreenDL considers a mobile edge network G that consists of 1) a set V of devices that are able to use device-to-device (D2D) communication with a *limited*, *discrete* set of possible transmission power levels $\mathcal{P}$ and 2) a set E of all possible links between smart devices in V, where each possible link $e \in E$ has a minimum transmission power level $\tau(e)$.[5] In addition, a user-defined ratio $\varphi \in [0,1]$ is given to indicate a lower bound of the link cardinality (i.e., *link cardinality ratio*) to guarantee the training performance. Note that the notation table is shown in Appendix B, and the examples of GreenDL are presented in Appendix C of the supplementary material.

The *decision variables* in GreenDL are listed as follows:

1) the decision variable $x_{vp} \in \{0,1\}$ denotes whether device $v \in V$ sets the transmission power level power as $p \in \mathcal{P}$,
2) the decision variable $y_e \in \{0,1\}$ denotes whether link $e \in E$ is selected, and
3) the decision variable $z^d_{uv} \in \{0,1\}$ denotes whether the flow is steered along the link from u to v to build a path from r to $d \in V \setminus \{r\}$, where $u, v \in V$ and $u \neq v$.

The integer linear programming (ILP) is listed as follows. The objective (1) is to minimize the total power consumption, where $\mathcal{P}$ is the set of candidate power levels and x_{vp} is the decision variable. Note that the other decision variables y_e, z^d_{uv} are used to constrain x_{vp}, and thus they only appear in the constraints (2)-(6).

$$\underset{\substack{x_{vp}, y_e, z^d_{uv} \in \{0,1\},\\ \forall u,v \in V, u \neq v, \forall e \in E, \forall p \in \mathcal{P}, \forall d \in V \setminus \{r\}}}{\text{minimize}} \quad \sum_{v \in V} \sum_{p \in \mathcal{P}} x_{vp} \cdot p \tag{1}$$

subject to

$$\sum_{e \in E} y_e \geq \varphi |E|, \tag{2}$$

$$\sum_{p \in \mathcal{P}} x_{vp} = 1, \forall v \in V, \tag{3}$$

$$y_e \leq \sum_{p \in \mathcal{P}: p \geq \tau(e)} x_{vp}, \forall e \in E, \forall v \in V(e), \tag{4}$$

$$\sum_{u \in V} z^d_{ru} - \sum_{u \in V} z^d_{ur} = 1, \forall d \in V \setminus \{r\}, \tag{5a}$$

$$\sum_{u \in V} z^d_{du} - \sum_{u \in V} z^d_{ud} = -1, \forall d \in V \setminus \{r\}, \tag{5b}$$

$$\sum_{u \in V} z^d_{vu} - \sum_{u \in V} z^d_{uv} = 0, \forall v, d \in V \setminus \{r\}, v \neq d, \tag{5c}$$

$$y_e \geq z^d_{uv}, \forall e \in E, \forall u, v \in V(e), u \neq v, \forall d \in V \setminus \{r\}. \tag{6}$$

Constraint (2) ensures an adequate number of links to guarantee the training performance. Constraint (3) limits each

[5]To explore the intrinsic property of GreenDL, we assume that both two end devices u and v are required to set the adequate transmission power level, which is no less than the minimum transmission power level $\tau(e)$, for communications in this paper.

device to choose a configuration for transmission power. Constraint (4) makes both devices with a possible link e have to choose a transmission power level no less than the minimum transmission power $\tau(e)$ to cover each other in the transmission range, where $V(e)$ denote the two corresponding devices incident to link e. *Flow-based* constraints (5a)–(5c) force the induced topology by the selected links to be *connected* (detailed later in the following paragraph), where r is an arbitrary device selected from V. Constraint (6) unions all the selected links on the paths from r to every device $d \in V \setminus \{r\}$ to induce the connected topology.

It is worth noting[6] that constraints (5a)–(5c) aim to make every device $d \in V \setminus \{r\}$ have a path from r in the induced topology to ensure the *network connectivity*. Constraint (5a) assures that r has one *outgoing* link and no *incoming* link on a path between r and d. Constraint (5b) guarantees that d has no outgoing link and one incoming link on a path between r and d. Constraint (5c) promises that that every device on a path between r and d must have one outgoing and one incoming links, while the other devices must have no outgoing and incoming links.

C. GreenDL with Collision Awareness

Subsequently, we introduce GreenDL with collision awareness (GreenDL-CA) to shorten the training time while preventing devices from serious collisions (or conflicts) during the data exchange. *Note that GreenDL-CA is not a new collision-avoidance protocol.* In the literature, many collision-aware broadcast problems in wireless networks are modeled as 2-hop coloring problems [52]–[55]. Such problems aim to bound the degree of vertices to avoid serious transmission collisions for the devices that share the same communication medium. Therefore, following the above works, our problem GreenDL-CA also limits the maximum degree of each device to a specific parameter Δ such that the required time slots in the broad scheduling is bounded within $\Delta^2 + 1$, where the parameter Δ is a tunable parameter to limit the maximum number of nearby devices within the transmission range of each device. The base station can set a large Δ when the number of available transmission resources is sufficient.

To this end, we impose a new constraint (i.e., eq. (7)) to limit the number of nearby devices within the transmission range of each device.

$$x_{vp} \cdot (\Delta - \sum_{e \in E(v): p \geq \tau(e)} 1) \geq 0, \quad \forall v \in V, \forall p \in \mathcal{P}. \tag{7}$$

Intuitively, setting a lower Δ can mitigate the collisions between devices and shorten the completion time of each round, since each device has at most Δ neighbors for exchanging parameters. In contrast, the greater Δ is set up, the longer the devices may wait for the completion of communications in each round.

[6]For more detailed explanation of flow-based constraints, please refer to Appendix D of the supplementary material.

D. Hardness Results

Then, we provide the rigorous proof of NP-hardness for GreenDL and GreenDL-CA. At the very beginning, we introduce an NP-hard problem, the Minimum Power Symmetric Connectivity (MPSC) problem.

Definition 1. Given an undirected graph $G_M = \{V_M, E_M\}$, where each possible link $e \in E_M$ is associated with a minimum transmission power $\tau_M(e)$, the Minimum Power Symmetric Connectivity (MPSC) Problem [51] asks for the assignment of transmission power $\gamma_M(v)$ for each node $v \in V_M$ to construct a tree T_M spanning all nodes in V_M such that:

1) For each link e in the tree T_M, both its incident nodes should be assigned with a power of no less than $\tau_M(e)$;
2) The total power consumption of the nodes in the network (i.e., $\sum_{v \in V_M} \gamma_M(v)$) is minimized.

The MPSC problem is proved to be NP-hard [51]. In the following, we show that GreenDL is NP-hard by reducing MPSC to GreenDL. Remark that the NP-hard proof of GreenDL-CA is omitted due to the similarity and the page limit.

Theorem 1. *GreenDL is NP-hard.*

Proof. We prove the theorem with a polynomial-time reduction from MPSC to GreenDL. For any given MPSC instance I_M with a graph $G_M = \{V_M, E_M\}$, we construct a GreenDL instance I with a graph $G = \{V, E\}$, a link cardinality ratio φ, and a power set $\mathcal{P}$ as follows. First, we set $\varphi = \frac{|V_M|-1}{|E_M|}$ and $\mathcal{P} = \{\tau_M(e_M) | \forall e_M \in E_M\}$ in I. Then, we construct a device and add it to V of I for each node in V_M. For any link in E_M, we create a link and add it to E to connect the corresponding devices in V. Last, for each link e_1 in E_M and its correspondingly created link $e_2 \in E$, the minimum transmission power level of e_2 is set to that of e_1 (i.e., $\tau(e_2) = \tau_M(e_1)$). It is obvious that the reduction can be done in polynomial time. Moreover, the optimal solution of I_M can be transformed to the optimal solution of I, and vice versa, since 1) $\varphi = \frac{|V_M|-1}{|E_M|} = \frac{|V|-1}{|E|}$ and 2) the minimization of power consumption makes the optimal solutions of I_M and I both become a tree. Therefore, the theorem follows. ☐

IV. Algorithm Design

In this section, we first introduce two naive approaches as the baselines for GreenDL with illustrative examples in Section IV-A. The design of CoTRAIN is detailed in Section IV-B. Note that the examples for each phase in CoTRAIN are presented in Appendix C. Then, the approximation ratio of CoTRAIN is shown in Section IV-C. Lastly, the extension of CoTRAIN (i.e., CoTRAIN-CA) is introduced in Section IV-D.

A. Two Naive Approaches for GreenDL

Two naive extensions from the well-known Kruskal's algorithm for the minimum-cost spanning tree (MST) problem can be modified to solve GreenDL. They respectively follow the idea behind Kruskal's algorithm, termed KR1 and KR2. The details of the Kruskal's algorithm, KR1, and KR2 are concisely introduced as follows. Due to the page limit, the pseudocodes of KR1 and KR2 are presented in Appendices E and F of the supplementary material.

1) *The Kruskal's algorithm* is a greedy-based method to find a minimum-cost spanning tree (MST) T in a given edge-weighted graph. It iteratively selects the edge with minimum weight and adds this edge to T if doing so does not form a cycle in T until a tree spanning all the nodes is constructed [56].

2) *KR1* is an extension of Kruskal's algorithm for GreenDL. KR1 has two phases. KR1's first phase iteratively selects the link e with the lowest $\tau(e)$ and adds it to the solution if doing so forms no cycle in the beginning. KR1 turns into the second phase once a tree connecting all devices is constructed. Similar to the first phase, KR1's second phase continues adding the link with the lowest $\tau(e)$ iteratively to meet the link cardinality ratio (i.e., a user-defined ratio of the requested link cardinality to the total number of possible device pair). However, the second phase does not avoid forming a cycle while it aims to select adequate links.

3) *KR2* is the other extension of Kruskal's algorithm for GreenDL. KR2 works similarly as KR1. The only difference is that KR2 iteratively selects the link that *increases the least total power consumption*.

However, KR1 and KR2 both neglect the interplay among the three challenges described in Section I such that most links selected by both methods are rarely in dense areas, and thus may waste power to cover a few links.

B. Rationale to Design CoTRAIN and Methodology

To efficiently solve GreenDL, we design an approximation algorithm named CoTRAIN to carefully address the above challenges. Instead of selecting a *single* link with the lowest minimum transmission power level for each time, CoTRAIN takes a *forward view* to select *multiple links* that have the *lowest power consumption per link cardinality* (i.e., *niche link set*). In this way, CoTRAIN can avoid excessively increasing the transmission power of devices in sparse areas and significantly reduce the transmission power. It introduces *niche indicator* to evaluate link sets and iteratively find the niche link set (or a close one) to augment the links until the *link cardinality ratio* is met, and then imposes a MST for the *network connectivity*. Finally, it removes the redundant links to save more energy. The pseudocode of CoTRAIN is presented in Algorithm 1.

Specifically, let E_t be the unselected links in E after link selection of the t^{th} iteration and $E_0 = E$, initially. Finding the niche link set in the network $\{V, E_{t-1}\}$ at the t^{th} iteration is equal to solving the following integer programming (IP), where the *decision variables* (x^t, y^t) in (8) are akin to the decision variables (x, y) in GreenDL (1).

$$\underset{\substack{x^t_{vp}, y^t_e \in \{0,1\},\\ \forall v\in V, \forall p\in\mathcal{P}, \forall e\in E_{t-1}}}{\text{minimize}} \quad \frac{\sum_{v\in V}\sum_{p\in\mathcal{P}} x^t_{vp}\cdot p}{\sum_{e\in E_{t-1}} y^t_e} \tag{8a}$$

subject to

$$y^t_e \le \sum_{p\in\mathcal{P}:p\ge\tau(e)} x^t_{vp}, \forall e\in E_{t-1}, \forall v\in V(e), \tag{8b}$$

Algorithm 1 CoTRAIN

Input: A given network $G = (V, E)$, a set of power levels $\mathcal{P}$, the minimum transmission power level $\tau(e)$ for each link $e \in E$, link cardinality ratio φ, and a loss percentage ϵ.
Output: The set of selected links $\mathcal{E}$.

```
NL Selection
1: $\mathcal{L} \leftarrow \emptyset, E_0 \leftarrow E, t \leftarrow 1$;
2: while $|\mathcal{L}| < (1-\epsilon)\varphi|E|$ do ▷ To constantly check the number of selected links
3:     $\overline{X}^t, \overline{Y}^t \leftarrow$ LP_Solver($V, E_{t-1}, \mathcal{P}$) ; ▷ To obtain the optimal fractional solution (OFS) of LP (9)
4:     $c_t \leftarrow 2\lceil\log|E_{t-1}|\rceil - 1$;
5:     for each $i \in [0, c_t - 1]$ do
6:         $\mathcal{C}^t_i \leftarrow \{e \in E_{t-1} | \; 0.5^{(i+1)} < \overline{y}^t_e \le 0.5^i\}$; ▷ To partition the OFS into the specified groups
7:     $\mathcal{L}_t \leftarrow \arg\max_{\mathcal{C}^t_i : 0\le i\le c_t-1} \Lambda(\mathcal{C}^t_i)$, tie breaking arbitrarily, where $\Lambda(\mathcal{C}^t_i) = |\mathcal{C}^t_i| - \frac{2^i}{c_t+1}$;
8:     $E_t \leftarrow E_{t-1} \setminus \mathcal{L}_t$;
9:     $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}_t$;
10:    $t \leftarrow t + 1$;
NC Provision
11: $\mathcal{T} \leftarrow MST(G)$; ▷ To obtain minimum-cost spanning tree of $G$
12: $\mathcal{E} \leftarrow (E \cap \mathcal{T}) \cup \mathcal{L}$;
RL Deletion
13: $E_{ex} \leftarrow \emptyset$; ▷ The set of the examined link
14: while $|\mathcal{E}| > (1-\epsilon)\varphi|E|$ do
15:     $e \leftarrow \arg\max_{e'\in\mathcal{E}\setminus E_{ex}} \Phi(\mathcal{E}) - \Phi(\mathcal{E} \setminus \{e'\})$, tie breaking arbitrarily;
16:     $E_{ex} \leftarrow E_{ex} \cup \{e\}$;
17:     if $\mathcal{E} \setminus \{e\}$ is connected then
18:         $\mathcal{E} \leftarrow \mathcal{E} \setminus \{e\}$;
19: return $\mathcal{E}$;
```

$$\sum_{e\in E_{t-1}} y^t_e \ge 1. \tag{8c}$$

However, the above IP is still *non-trivial*. To this end, CoTRAIN subtly 1) *relaxes* the integral restriction of decision variables $x^t_{vp}, y^t_e \in \{0,1\}$ in (8) to the fractional restriction of the *decision variables* $\overline{x}^t_{vp}, \overline{y}^t_e \ge 0$ and 2) *scales* $\sum_{e\in E_{t-1}} y^t_e \ge 1$ to $\sum_{e\in E_{t-1}} \overline{y}^t_e = 1$ to get a *linear programming* (LP) as follows.

$$\underset{\substack{\overline{x}^t_{vp}, \overline{y}^t_e \ge 0,\\ \forall v\in V, \forall p\in\mathcal{P}, \forall e\in E_{t-1}}}{\text{minimize}} \quad \sum_{v\in V}\sum_{p\in\mathcal{P}} \overline{x}^t_{vp}\cdot p \tag{9a}$$

subject to

$$\overline{y}^t_e \le \sum_{p\in\mathcal{P}:p\ge\tau(e)} \overline{x}^t_{vp}, \forall e\in E_{t-1}, \forall v\in V(e), \tag{9b}$$

$$\sum_{e\in E_{t-1}} \overline{y}^t_e = 1. \tag{9c}$$

In this way, the *fractional optimal solution* $(\overline{x}^t, \overline{y}^t)$ can be acquired by a polynomial-time LP solver (e.g., Gurobi [57]), and CoTRAIN can find a near-optimal niche link set (i.e., close to the niche link set) based on the *clue* given by $(\overline{x}^t, \overline{y}^t)$.

To acquire the near-optimal niche link set, CoTRAIN constructs *multiple* candidate niche link sets in each iteration and chooses the one with the *largest* niche indicator. Specifically, at the t^{th} iteration, CoTRAIN constructs c_t candidate niche link sets, $\mathcal{C}_0^t$, $\mathcal{C}_1^t$, ..., $\mathcal{C}_{c_t-1}^t$, and the i^{th} candidate is

$$\mathcal{C}_i^t = \{e \in E_{t-1} |\ 0.5^{(i+1)} < \overline{y}_e^t \leq 0.5^i\}, \tag{10}$$

where $c_t = 2\lceil \log |E_{t-1}| \rceil - 1$ and $0 \leq i \leq c_t - 1$. Then, the niche indicator of each niche link set $\mathcal{C}_i^t$ is defined as

$$\Lambda(\mathcal{C}_i^t) = |\mathcal{C}_i^t| - \frac{2^i}{c_t + 1}. \tag{11}$$

Later we show that the *niche link set* and *niche indicator* are the cornerstones of CoTRAIN to ensure the approximation ratio.

CoTRAIN includes the following three phases: 1) Niche Link Selection (NL Selection), 2) Network Connectivity Provision (NC Provision), and 3) Redundant Link Deletion (RL Deletion). Particularly, NL Selection first iteratively constructs multiple candidate niche link sets, chooses the best set among them, and adds the links of the chosen set into the selected links until the link cardinality ratio is satisfied. NC Provision then imposes a MST to connect all devices to guarantee network connectivity. Finally, RL Deletion eliminates redundant links to reduce total power consumption. To achieve the approximation ratio, it is important for NL Selection to evaluate the constructed candidate niche link sets by niche indicator to find the near-optimal niche link set.

1) Niche Link Selection (NL Selection): NL Selection *iteratively* finds the near-optimal niche link set and adds the links of the set to the selected links until the link cardinality ratio is satisfied. Recall that E_t is the unselected link set in E after link selection of the t^{th} iteration and $E_0 = E$, initially. For the t^{th} iteration, NL Selection obtains the optimal fractional solution $(\overline{x}^t, \overline{y}^t)$ of LP (9) with the network[7] $\{V, E_{t-1}\}$ by any existing LP solver (e.g., Gurobi [57]). Then, it constructs c_t *candidate niche link sets* according to eq. (10) and chooses the set with the largest niche indicator (see eq. (11)) among the c_t candidate sets, where $c_t = 2\lceil \log |E_{t-1}| \rceil - 1$. The candidate niche link set chosen in the t^{th} iteration can be written as

$$\mathcal{L}_t = \underset{\mathcal{C}_i^t : 0 \leq i \leq c_t - 1}{\arg\max}\ \Lambda(\mathcal{C}_i^t). \tag{12}$$

Afterward, NL Selection adds the links in $\mathcal{L}_t$ into the selected links, and then employs the sub-network with $\{V, E_t\}$, where $E_t \leftarrow E_{t-1} \setminus \mathcal{L}_t$, for the next iteration (i.e., the $(t+1)^{th}$ iteration) to compute the optimal fractional solution of LP (9) and $\mathcal{L}_{t+1}$. NL Selection repeats link selection until the number of selected links is at least $(1-\epsilon)\varphi|E|$, where ϵ is a positive tunable parameter to limit the loss percentage of requested link cardinality. Now, the link set selected by NL Selection is $\mathcal{L} = \bigcup_{t=1}^k \mathcal{L}_t$, where k is the number of executed iterations.

2) Network Connectivity Provision (NC Provision): NC Provision employs Kruskal's algorithm to get a MST $\mathcal{T}$ of the network G to ensure the *network connectivity*. To precisely calculate the power consumption, each link e is associated with a new cost $\phi(e) = \min_{p \in \mathcal{P}: p \geq \tau(e)} p$ for computing $\mathcal{T}$. After NC Provision, the selected link set becomes $(E \cap \mathcal{T}) \cup \mathcal{L}$.

3) Redundant Link Deletion (RL Deletion): The final phase eliminates redundant links so as to further decrease the power consumption. Let $\mathcal{E}$ and $\Phi(\mathcal{E})$ denote the current selected links (i.e., $\mathcal{E} \leftarrow (E \cap \mathcal{T}) \cup \mathcal{L}$) and its transmission power consumption. Since each device in the network should have an adequate transmission power level,

$$\Phi(\mathcal{E}) = \sum_{v \in V} \max_{e \in E(v) \cap \mathcal{E}} \phi(e), \text{ where } \phi(e) = \min_{p \in \mathcal{P}: p \geq \tau(e)} p, \tag{13}$$

and $\mathcal{E}$ denotes the selected links. RL Deletion iteratively removes the link e, the removal of which saves the most power until $|\mathcal{E}| \leq \varphi|E|$, i.e.,

$$e = \underset{e \in \mathcal{E}}{\arg\max}\, \Phi(\mathcal{E}) - \Phi(\mathcal{E} \setminus \{e\}). \tag{14}$$

However, to keep the *network connectivity*, RL Deletion does not remove link e if removing e splits the induced topology.

C. Theoretical Analysis

We first analyze the time complexity of CoTRAIN.

Time Complexity 1. *Let T_{LP} denote the time complexity of solving LP (9). The time complexity of CoTRAIN is $O(T_{LP}|E|)$.*

Proof. NL Selection in CoTRAIN performs at most $|E|$ times of T_{LP} to cover links so the time complexity in NC Provision is $O(T_{LP}|E|)$. NC Provision in CoTRAIN performs Kruskal's algorithm to get a MST so its time complexity is $O(|E| \log |V|)$. Lastly, RL Deletion checks redundant at most $|E|$ links for $|E|$ links so its time complexity is $O(|E|^2)$. Therefore, the time complexity of CoTRAIN is dominated by NL Selection with time complexity of $O(T_{LP}|E|)$. □

Remark that the LP (9) can be solved in polynomial time, and the best known complexity of LP solver in the literature is $O(\alpha^{2.37} \log(\alpha/\beta))$, where α is the number of decision variables and β is the bit-complexity [58]. Recall that the LP (9) has $|V| \cdot |\mathcal{P}| + |E| = O(|E|)$ decision variables at the t^{th} iteration (i.e., $\overline{x}_{vp}^t$ and $\overline{y}_e^t$) since $|\mathcal{P}|$ is usually a small constant and $|V| \leq |E|$. Thus, the complexity of solving the LP (9) is $O(|E|^{2.37} \log |E|)$, and the overall time complexity of CoTRAIN is $O(|E|^{3.37} \log |E|)$.

To prove that CoTRAIN guarantees an approximation ratio (i.e., Theorem 2), we first claim Lemmas 1, 2, and 3 as follows. Note that Lemma 2 holds if Lemma 1 holds as well. Then, with Lemmas 2 and 3, Theorem 2 immediately follows. For ease of reading, the proofs of Lemmas 1, 2, and 3 are provided in Appendix G of the supplementary material.

Lemma 1. *At the t^{th} iteration of NL Selection, the chosen near-optimal niche link set $\mathcal{L}_t$ has the power consumption per link cardinality $\Phi(\mathcal{L}_t)/|\mathcal{L}_t|$ less than $\lceil \log |E_{t-1}| \rceil \cdot \Phi(\mathcal{L}_t^*)/|\mathcal{L}_t^*|$, where $\mathcal{L}_t^*$ denotes the optimal niche link set at the t^{th} iteration, i.e., the optimal solution of IP (8).*

Lemma 2. *NL Selection outputs a set of links $\mathcal{L}$ with the power consumption $\Phi(\mathcal{L}) < \lceil \log |E| \rceil (1 + \ln \frac{1}{\epsilon} +$*

[7] Note that when the number of unselected links is smaller than nine, i.e., $|E_{t-1}| < 9$, NL Selection can compute the optimal niche link set since it only has to examine at most $2^8 = 256$ possibilities, which takes a small constant time. Therefore, it suffices to deal with the cases where $|E_{t-1}| \geq 9$.

$\frac{1-\varphi}{\epsilon\varphi})\Phi(OPT)$ *while ensuring* $|\mathcal{L}| \geq (1-\epsilon)\varphi|E|$, *where* OPT *is the optimal solution of GreenDL and* ϵ *is a positive tunable parameter to limit the loss percentage of requested link cardinality.*

Lemma 3. *The power consumption of MST constructed by NC Provision is* $\Phi(\mathcal{T}) \leq 2 \cdot \Phi(OPT)$.

Theorem 2. *CoTRAIN is a* $(O(\log|E|(\ln\frac{1}{\epsilon}+\frac{1-\varphi}{\epsilon\varphi})), 1-\epsilon)$-*approximation algorithm, where* ϵ *is a positive tunable parameter to limit the loss percentage of requested link cardinality.*

Proof. Lemmas 2 indicates the gap between the optimal solution and the intermediate solution generated by NL Selection. Lemma 3 further shows the gap between the optimal solution and the MST generated by NC Provision. Combining Lemmas 2 and 3, CoTRAIN outputs the link set $\mathcal{E}$ after NC Provision. Thus, the power consumption of the solution $\mathcal{E}$ is

$$\Phi(\mathcal{E}) < (2 + \lceil\log|E|\rceil(1+\ln\frac{1}{\epsilon}+\frac{1-\varphi}{\epsilon\varphi}))\cdot\Phi(OPT). \quad (15)$$

Finally, the theorem follows since link deletion by RL Deletion does not increase the power consumption and it stops once $|\mathcal{E}| \leq \varphi|E|$, i.e., $|\mathcal{E}|$ is still at least $(1-\epsilon)\varphi|E|$. □

D. CoTRAIN with Collision Awareness (CoTRAIN-CA)

We extend CoTRAIN to CoTRAIN-CA for GreenDL-CA as follows. Recall that the first phase, NL Selection, iteratively constructs the candidate niche link sets (i.e., the sets of links for selection in each iteration) by solving the following linear programming (LP) (9) and selects the one with the largest niche indicator among the constructed candidate niche link sets. To support GreenDL-CA, NL Selection is modified to adopt LP (9) combined with eq. (16) to limit the number of nearby devices within the transmission range of each device.

$$\overline{x}^t_{vp}\cdot(\Delta - \sum_{e\in E(v):p\geq\tau(e)} 1) \geq 0, \quad \forall v\in V, \forall p\in\mathcal{P} \quad (16)$$

Subsequently, the second phase, NC Provision, is modified to construct a MST without selecting any link that violates eq. (7). It then adds such a tree to the solution to connect all devices for the network connectivity. Finally, to further remedy the collision issue, we modify the procedure in the third phase, RL Deletion, where the rationale is to make it tend to prune the links that connect the devices with more neighbors. More specifically, the modified RL Deletion follows the original one to iteratively remove a link until $|\mathcal{E}| \leq \varphi|E|$. The difference is that RL Deletion gives the priority to the links whose corresponding devices have the more neighbors, and then selects the link that saves the most power among them. Note that the results of Theorem 2 still holds for CoTRAIN-CA due to the similarity. The pseudocode of CoTRAIN-CA is presented in Algorithm 2.

Algorithm 2 CoTRAIN-CA

```
Input: A given network $G=(V,E)$, a set of power levels $\mathcal{P}$, the
    minimum transmission power level $\tau(e)$ for each link $e\in E$,
    link cardinality ratio $\varphi$, and a loss percentage $\epsilon$.
Output: The set of selected links $\mathcal{E}$.
NL Selection
 1: $\mathcal{L}\leftarrow\emptyset, E_0\leftarrow E, t\leftarrow 1$;
 2: while $|\mathcal{L}| < (1-\epsilon)\varphi|E|$ do  ▷ To constantly check the number of selected links
 3:     $\overline{X}^t, \overline{Y}^t \leftarrow$ LP_Solver($V, E_{t-1}, \mathcal{P}$) ;  ▷ To obtain the optimal fractional solution (OFS) of LP (9) with eq. (16)
 4:     $c_t \leftarrow 2\lceil\log|E_{t-1}|\rceil - 1$;
 5:     for each $i\in[0, c_t-1]$ do
 6:         $\mathcal{C}^t_i \leftarrow \{e\in E_{t-1} |\ 0.5^{(i+1)} < \overline{y}^t_e \leq 0.5^i\}$;  ▷ To partition the OFS into the specified groups
 7:     $\mathcal{L}_t \leftarrow \arg\max_{\mathcal{C}^t_i:0\leq i\leq c_t-1}\Lambda(\mathcal{C}^t_i)$, tie breaking arbitrarily, where $\Lambda(\mathcal{C}^t_i) = |\mathcal{C}^t_i| - \frac{2^i}{c_t+1}$;
 8:     $E_t \leftarrow E_{t-1}\setminus\mathcal{L}_t$;
 9:     $\mathcal{L}\leftarrow\mathcal{L}\cup\mathcal{L}_t$;
10:     $t\leftarrow t+1$;
NC Provision
11: $G\leftarrow G\setminus\{e\in E|\ \tau(e) > \omega(v), \exists v\in V(e)\}$, where $\omega(v) = \max\{p\in\mathcal{P}|\ \Delta\geq\sum_{e'\in E(v):p\geq\tau(e')}1\}$;
12: $\mathcal{T}\leftarrow MST(G)$; ▷ To obtain minimum-cost spanning tree of $G$
13: $\mathcal{E}\leftarrow(E\cap\mathcal{T})\cup\mathcal{L}$;
RL Deletion
14: $E_{ex}\leftarrow\emptyset$;
15: while $|\mathcal{E}| > (1-\epsilon)\varphi|E|$ do
16:     $V_{hi}\leftarrow\{v_1\in V|\ |E(v_1)\cap\mathcal{E}\setminus E_{ex}| = \max_{v_2\in V}|E(v_2)\cap\mathcal{E}\setminus E_{ex}|\}$;  ▷ To obtain the devices $V_{hi}$ with the highest degree in $\mathcal{E}$
17:     $e \leftarrow \arg\max_{e'\in\cup_{v\in V_{hi}}\delta(v)\setminus E_{ex}}\Phi(\mathcal{E}) - \Phi(\mathcal{E}\setminus\{e'\})$, tie breaking arbitrarily; ▷ To obtain the link of $V_{hi}$ that reduces the most power consumption
18:     $E_{ex}\leftarrow E_{ex}\cup\{e\}$;
19:     if $\mathcal{E}\setminus\{e\}$ is connected then
20:         $\mathcal{E}\leftarrow\mathcal{E}\setminus\{e\}$;
21: return $\mathcal{E}$;
```

V. Performance Evaluation

In this section, we first describe the experimental settings (Section V-A) and then evaluate the power consumption of CoTRAIN/CoTRAIN-CA, KR1/KR1-CA, and KR2/KR2-CA. Lastly, we implement DL using the sets of links selected by the six methods above. Overall, the extensive simulations and implementation cover the following aspects:

1) *Practices in Real-World Networks*. We show that CoTRAIN and CoTRAIN-CA can be practically applicable to the scenarios and save a great deal of power in real-world networks by simulations (Section V-B).

2) *Scalability in Synthetic Networks*. With varying parameters δ, n, ρ, and Δ, we evaluate the performance of each method and explore the findings of the characteristics of KR1 and KR2 (Section V-C).

3) *Implementation of DL*. We evaluate the decentralized training of neural network models with the set of links selected by different methods and further examine the impact posed by the non-IID data partitions (Section V-D).

A. Experimental Settings

1) Network Magnitude:

• To take the real-world D2D transmission into account, we evaluate three methods in two networks with 16 and 32 devices extracted from Santander City [16] and we refer to the two networks as *the small real-world network* (with 16 devices) and *the median real-world network* (with 32 devices), respectively.

• In order to examine the performance of three methods in larger networks, we also expand the network magnitude (i.e., number of devices) n and randomly deploy the devices in a two-dimensional 50×50 m^2 area (i.e., not extracted from Santander City). We refer to the randomly-deployed networks as *synthetic networks*.

2) Transmission Configurations: We assume that there are only *five available configurations* of transmission power (i.e., $|\mathcal{P}| = 5$) for D2D broadcast transmission. In the real-world network dataset of Santander City [16], each device has a few neighboring devices within the range of 30 m such that even constructing a connected topology for the devices is difficult. Therefore, in this paper, the minimum link range of D2D is set to 30 m, and the corresponding minimum transmission power is $(0.1 \times 50^{-4}) \times 30^4 \times (1 + 0.5) \approx 0.02$ mW according to the Distance-Based Path-Loss Power Control (DPPC) scheme [59], where the receiver sensitivity, path-loss exponent, and estimation margin are set to (0.1×50^{-4}), 4, and 0.5, respectively. Similarly, the maximum link range of D2D is set to 100 m, and thus the corresponding minimum transmission power is $(0.1 \times 50^{-4}) \times 100^4 \times (1 + 0.5) \approx 2.4$ mW. The parameters for applying the DPPC scheme are referred to the settings in [59].

3) Varying Parameters: We vary four variables to observe the power consumption of the three methods, which are i) the *demand average degree* δ (i.e., average number of links per device), ii) the *network magnitude* n, iii) the *device density* ρ (i.e., number of devices per unit of area), and iv) the *maximum degree constraint*[8] Δ, respectively.

• The small and median real-world networks: We alter the values of δ ranging from 4 to 12 and set Δ to be n and 2δ for GreenDL and GreenDL-CA, respectively, since the other two variables are predetermined (i.e., $n = 16$ or 32, and ρ is determined by the random extraction from Santander City). To see the optimality gap between three methods and the optimal solutions, denoted by the OPT (i.e., the minimum power consumption to satisfy the requirement of link cardinality), we used Gurobi [57] to obtain the OPT. Each result of experiments is averaged over 500 times, and the confidence level is 95%.

• The synthetic networks: We conduct extensive experiments of three methods by altering one of the four variables and fixing the others. The default values of four variables are 500 devices, 10 links/device, 0.005 device$/m^2$, and 24 (or n if the maximum degree constraint is not imposed), respectively. Each result of experiments is averaged over 500 times and the confidence level is 95%.

4) The DL Implementation Detail:

• We adopt the well-known dataset CIFAR10 [13] with 50,000 images.

• The input images for training are preprocessed according to [18]. TensorFlow and Keras are used to implement a convolutional neural network (CNN), which has two convolutional layers (CL) and three fully connected layers (FL). Both two CLs have 64 channels and each layer is followed by a 3×3 max pooling with a stride of two and normalization. The first two FLs have 384 and 192 units (each of them with ReLu activation followed by one dropout), and the last FL is the final softmax output layer with 10 units. We adopted the optimizer of mini-batch gradient descent [60]. The learning rate, learning rate decay, number of local epochs, and local minibatch size are set to 0.2, 0.99, 1, and 64, respectively.

• To implement IID and non-IID DL, we follow the method of data partitions in [17] to present the *degree of data skewness*. We control the *skewness* $\mathcal{K} \in [0, 1]$ by assigning distinct fractions of non-IID data to each device. For example, if $\mathcal{K} = 0.2$, each device is allocated with a data partition, 20% of which belongs to the same class and 80% of which is IID.

• We train the model described above with different topologies built by three methods and with different degrees of $\mathcal{K}$ to examine the performance of CoTRAIN. We adopt two networks with 16 and 32 devices to examine the performance, both of which are extracted randomly in Santander City. Each result of training is averaged over 20 times and the confidence level is 90%. The training process of DL is implemented in and emulated by an HP Z8G4 server with two Intel Xeon Silver 4114 CPUs and two GPUs of GeForce RTX 3090.

B. Practices in Real-World Networks

The results of three methods and the OPT with and without considering collision awareness are shown in Fig. 2. The gap between the results of CoTRAIN and CoTRAIN-CA and the OPT are bounded. Although the results of KR1/KR1-CA and KR2/KR2-CA are close to those of CoTRAIN/CoTRAIN-CA when the number of devices is 16, as shown in Figs. 2(a) and 2(c), the broader difference emerges when the network magnitude doubles and CoTRAIN/CoTRAIN-CA still closely stick to the OPT, as depicted in Figs. 2(b) and 2(d). It is because CoTRAIN/ CoTRAIN-CA take a forward view of possible sets of links (i.e., niche indicator and niche link set) while both KR1/KR1-CA and KR2/KR2-CA look for one best possible link alone to cover. The running time of different algorithms is shown in Table I. Gurobi takes more than 3.6 hours to derive the OPT solution for GreenDL as the number of devices is 100, and its running time increases drastically along with the increase of numbers of devices. In contrast, the other three algorithms including CoTRAIN increases the running time smoothly and only compute within 1 second as the number of devices is 100. The result indicates that an approximation solution is indispensable.

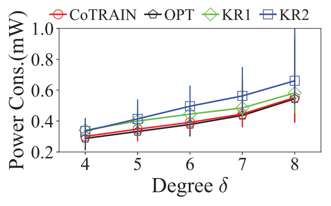
C. Scalability in Synthetic Networks

First, we investigate the results of three methods without considering collision awareness. Overall, power consumption increases when δ and n go up as shown in Fig. 3. KR1 selects the links with lowest required power yet ignores discrete power levels and symmetric wireless links. The selected links tends to be sparse and waste much energy of devices to meet the power level for connecting devices. KR2 selects the links

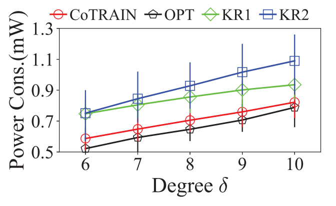
[8]Recall that in Section III-C, Δ represents the maximum number of nearby devices within the transmission range of each device. Therefore, setting a lower Δ can mitigate the collisions among devices. In contrast, $\Delta = n$ indicates that the maximum degree is not restricted.

TABLE I
RUNNING TIME OF DIFFERENT ALGORITHMS (SEC)

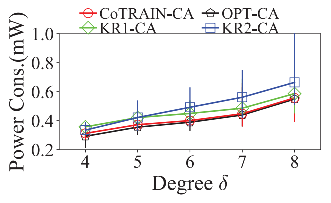
Algorithm	CoTRAIN	KR1	KR2	OPT
60 devices	0.0956 (1x)	0.0018 (0.02x)	0.0115 (0.12x)	73.611 (769.71x)
70 devices	0.1151 (1x)	0.0026 (0.02x)	0.0172 (0.15x)	175.765 (1527.62x)
80 devices	0.1474 (1x)	0.0035 (0.02x)	0.0249 (0.17x)	725.806 (4924.02x)
90 devices	0.2170 (1x)	0.0049 (0.02x)	0.0346 (0.16x)	3792.81 (17477.18x)
100 devices	0.3771 (1x)	0.0062 (0.02x)	0.0477 (0.13x)	13249.8 (35137.25x)



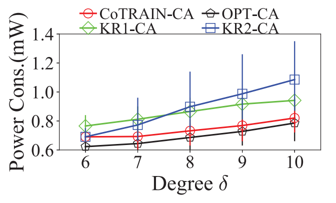
(a) Small real-world network (16 devices)



(b) Median real-world network (32 devices)

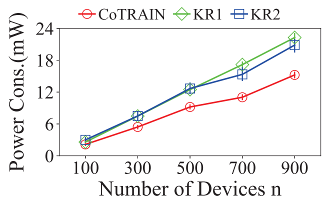


(c) Small real-world network with collision awareness

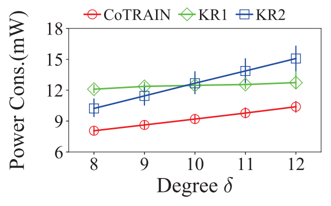


(d) Median real-world network with collision awareness

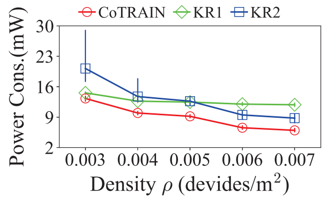
Fig. 2. Effect of different degree and collision awareness on power consumption.



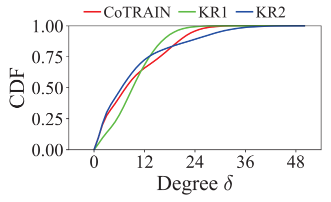
(a) Num. of devices vs power cons.



(b) Degree vs power cons.

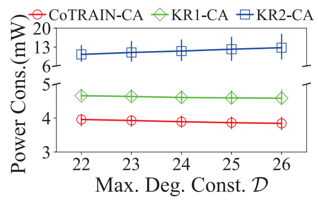


(c) Density vs power cons.

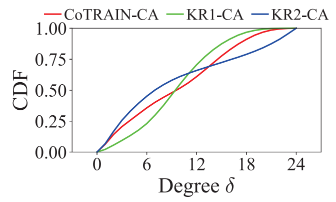


(d) CDF of device degree

Fig. 3. Effect of different parameters in large synthesis networks.



(a) Max. deg. constraint vs power cons.



(b) CDF of device degree with max. deg. constraint

Fig. 4. Effect of maximum degree constraint in large synthesis networks.

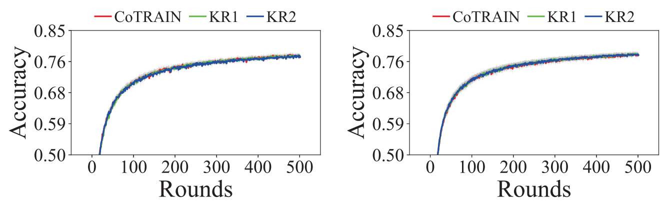
with the lowest increase on overall power consumption. Some devices may be forced to use the maximum power level to cover links. In contrast, the links of each candidate niche link set in CoTRAIN tend to be energy-saving. Then, by niche indicator, CoTRAIN can choose the set with more links to approximate the optimal solution.[9] To observe the growth of degree of each device in the established topology, Fig. 3(d) shows the distribution of device degree. KR1 tends to select links uniformly in the network since it overlooks the relation among links. More than 70% of devices have $5 \sim 20$ neighbors whereas those devices only account for 40% in KR2 and CoTRAIN. KR2 selects the links with lowest increase of power consumption such that it has the most devices with more than 30 selected links. When n is high, δ is low, or ρ is high, KR2 outperforms KR1 since more links for selection highlight the importance of relation among links as show in Figs. 3(a), 3(b), and 3(c). However, none of them can always generate desired solutions. By contrast, CoTRAIN can always balance the two factors to reduce the power consumption.

Then, we examine the performance of three methods with the maximum degree constraint. The results are shown in Fig. 4. We can see that the performance of KR2-CA deviates drastically due to the feature of KR2-CA (i.e., selecting links increasing the least power consumption) and the maximum degree constraint. The CDF of degree of the three methods in Fig. 4(b) follows the similar growth rate of that in Fig. 3(d). On the contrary, the results of both KR1-CA and CoTRAIN-CA are less deviated. However, CoTRAIN-CA can attain much lower power consumption since CoTRAIN-CA subtly overcomes the three challenges even though the maximum degree constraint is imposed. In conclusion, CoTRAIN/CoTRAIN-CA exploit the advantages and evade the disadvantages of the others. In spite of the imposition of maximum degree constraint, CoTRAIN-CA can still outperform the others by at least 20% of power consumption.

[9]The optimal solution of GreenDL is obtained only for the small and median networks since it is NP-hard and the running time is exponential to the network size.

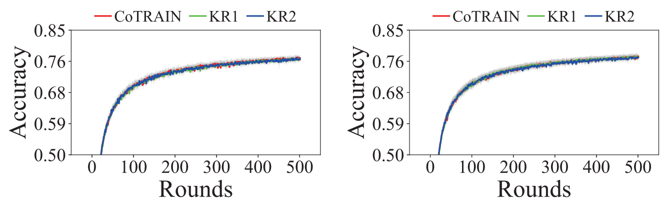
D. Implementation of DL

The results of different degrees of skewness, where $\mathcal{K}$ = 0%, 20%, 40%, 60%, and 80%, are shown in Figs. 5, 6, 7, 8,



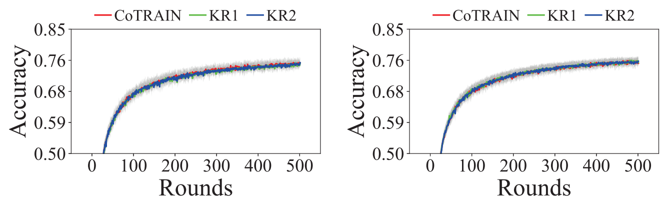
(a) 16 devices (b) 32 devices

Fig. 5. $\mathcal{K} = 0\%$ (Uniform data distribution)



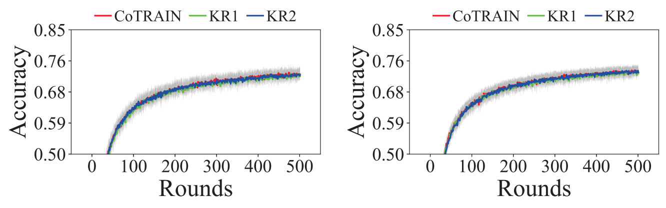
(a) 16 devices (b) 32 devices

Fig. 6. $\mathcal{K} = 20\%$ (i.e., 20% of data belongs to one class)



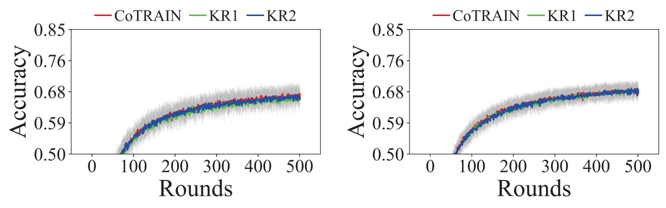
(a) 16 devices (b) 32 devices

Fig. 7. $\mathcal{K} = 40\%$ (i.e., 40% of data belongs to one class)



(a) 16 devices (b) 32 devices

Fig. 8. $\mathcal{K} = 60\%$ (i.e., 60% of data belongs to one class)



(a) 16 devices (b) 32 devices

Fig. 9. $\mathcal{K} = 80\%$ (i.e., 80% of data belongs to one class)

and 9, respectively. We omit the results with $\mathcal{K} = 100\%$ (i.e., each device is allocated one class of training data) since the model does not converge.[10] *Note that the deviation areas (i.e., the deviated accuracy due to mini-batch gradient descent) of three methods are highly overlapping and rendered in gray.* It is obvious to see that the degree of skewness poses great impact on the accuracy and convergence rate in the networks with different magnitude for three methods. Remark that the performance deviates severely when $\mathcal{K}$ gets higher. This is because the data skewness broadens the divergence of stochastic gradient descent of each batch. Three methods achieve the similar accuracy when using the almost identical numbers of rounds, which results from the same link cardinalities. However, as shown in Figs. 2, 3, and 4, CoTRAIN demands far less power consumption than the other two *in each round of training*. The cumulative power consumption of three methods for the DL implementation, with $\mathcal{K} = 0\%, n = 16$ and 32 (extracted from Santander City), is presented in Table II. Simply put, CoTRAIN conserves more than 20% of power consumption for transmission compared to the other two when the training process converges (i.e., achieving converged accuracy of 78%).

In addition, to show that the degree bound constraint can efficiently reduce the collisions, we conduct the following experiment. Assume that the model size is 4 MB, the number of available resource blocks is 1, and the bandwidth is 180 KHz. We set the transmission power according to the Distance-Based Path-Loss Power Control (DPPC) scheme [59] to ensure that the SINR within the desired link range is at least 10 dB. Thus, the transmission rate is $180 \times 1000 \times \log_2(1 + 10) = 621000$ bps $= 0.621$ mpbs by the Shannon capacity. Therefore, each device's transmission time in a training round is about $4 \times 8/0.621 \approx 51.5$ seconds. The results of transmission time are summarized in Table III. CoTRAIN-CA takes less time than CoTRAIN for all settings of δ. It is because CoTRAIN-CA sets Δ to 2δ to limit the maximum number of 2-hop neighbors of a device such that more devices can broadcast at the same time with the power level properly allocated.

In summary, we empirically show that CoTRAIN and CoTRAIN-CA are effective in conserving power consumption in the practical DL implementation without compromising accuracy.

VI. Conclusions and Future Works

This paper studies a new optimization problem, GreenDL, to explore the trade-off between transmission power consumption and training performance in depth. In addition, GreenDL is extended to be GreenDL-CA so as to mitigate the issue of collision and interference by means of limiting the maximum degree of each device. We rigorously prove the NP-hardness of GreenDL and GreenDL-CA and explore three new challenges derived from both, which are discrete power levels, symmetric wireless links, and density-aware power selection. For GreenDL, we propose a novel algorithm CoTRAIN to subtly take a *forward view* to selects links sets by exploiting

[10]The approach to mitigating the influence of non-IID data is beyond the scope of this paper. Please refer to [17], [27] for more details.

TABLE II
TRANSMISSION POWER CONSUMPTION IN THE SMALL AND MEDIAN REAL-WORLD NETWORKS (MW)

Accuracy	69%	72%	75%	78%
CoTRAIN-16	32.89 (1x)	48.56 (1x)	81.45 (1x)	170.73 (1x)
KR1-16	36.05 (1.10x)	57.86 (1.19x)	96.14 (1.18x)	214.54(1.26x)
KR2-16	40.70 (1.24x)	61.54 (1.27x)	113.16 (1.39x)	234.27 (1.37x)
CoTRAIN-32	50.79 (1x)	79.01 (1x)	130.50 (1x)	279.34 (1x)
KR1-32	64.21 (1.26x)	96.74 (1.22x)	162.66 (1.25x)	336.45 (1.20x)
KR2-32	67.76 (1.33x)	108.60 (1.37x)	178.22 (1.37x)	386.15 (1.38x)

TABLE III
TOTAL TRANSMISSION TIME WITH 16 DEVICES (MIN)

Accuracy	65%	68%	72%	74%
CoTRAIN-δ = 4	909.5(1x)	1955.0(1x)	4139.5(1x)	-
CoTRAIN-CA-δ = 4	493.9(0.5x)	731.9(0.4x)	2546.6(0.6x)	-
CoTRAIN-δ = 5	688.5(1x)	1122.0(1x)	2524.5(1x)	-
CoTRAIN-CA-δ = 5	464.1(0.8x)	714.0(0.7x)	1666.0(0.7x)	-
CoTRAIN-δ = 6	724.2(1x)	979.2(1x)	2560.2(1x)	3723.0(1x)
CoTRAIN-CA-δ = 6	596.7(0.8x)	826.2(0.8x)	1698.3(0.7x)	2968.2(0.8x)
CoTRAIN-δ = 7	856.8(1x)	1082.9(1x)	1749.3(1x)	2499.0(1x)
CoTRAIN-CA-δ = 7	569.5(0.7x)	850.0(0.8x)	1419.5(0.8x)	1895.5(0.8x)
CoTRAIN-δ = 8	868.7(1x)	1225.7(1x)	2463.3(1x)	4046.0(1x)
CoTRAIN-CA-δ = 8	617.1(0.7x)	813.5(0.7x)	1393.2(0.6x)	2094.4(0.5x)

insightful *niche indicator* to assess link sets. For GreenDL-CA, we propose CoTRAIN-CA, which reserves the advantages of CoTRAIN in spite of the constraint of the maximum degree of each device. We show that both CoTRAIN and CoTRAIN-CA achieve a logarithmic approximation ratio and polynomial time complexity. On the implementation side, we evaluate the performance of CoTRAIN and CoTRAIN-CA threefold compared to two naive Kruskal's-algorithm-based approaches, KR1 and KR2. The results show that both CoTRAIN and CoTRAIN-CA outperform the others by at least 20% with no compromise of performance in practical DL.

The main contributions of this paper focus on *exploring the feasibility of the energy-efficient decentralized learning based on the construction of topology for smart devices via D2D communications*. To the best of our knowledge, this potential and interesting topic has not been explored in depth before. To explore the intrinsic feasibility, it is nature to study a relatively *simplified* formulation, that is, *exploiting a large number of links* in communication topology leads to better convergence rate in average. However, a larger number of links does not always lead to a smaller training time in realistic scenarios. The first reason is that selecting more links requires the devices to set a higher transmission power level, and thus it may cause more conflicts among the devices and prolong the communication time in each training round. The second reason is that the spectral gap and the hitting time also affect the number of training rounds [28], [61]. Therefore, it will be an interesting subject to explore a more complicated trade-off between the energy cost and training time from multiple perspectives at the same time.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 108-2221-E-194-025-MY3 and MOST 107-2218-E-194-016-MY3; and in part by the Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI) from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

REFERENCES

[1] C. W. Ching, C. K. Yang, Y. C. Liu, C. W. Hsu, J. J. Kuo, H. S. Huang, and J. F. Lee, "Energy-efficient link selection for decentralized learning via smart devices with edge computing," in *Proc. IEEE GLOBECOM*, 2020, pp. 1–6.
[2] C. Zhang *et al.*, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 2224–2287, 2019.
[3] T. Yang *et al.*, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.
[4] (2020) Cisco annual internet report (2018–2023). White paper.
[5] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. PMLR ICML*, 2019, pp. 3478–3487.
[6] Y. Li *et al.*, "Pipe-SGD: A decentralized pipelined SGD framework for distributed deep net training," in *Proc. NeurIPS*, 2018, pp. 8056–8067.
[7] S. Shi, X. Chu, and B. Li, "MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms," in *Proc. IEEE INFOCOM*, 2019, pp. 172–180.
[8] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. NeurIPS*, 2018, pp. 1306–1316.
[9] D. Alistarh *et al.*, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NeurIPS*, 2017, pp. 1707–1718.
[10] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning (early access)," *IEEE Trans. Intell. Transp. Syst.*, 2020.
[11] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1030–1040, 2020.

[12] Y. He, J. Ren, G. Yu, and Y. Cai, "Joint computation offloading and resource allocation in D2D enabled mec networks," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, University of Toronto*, 2009.

[14] Q. Ju *et al.*, "Predictive power management for internet of battery-less things," *IEEE Trans. Power Electron.*, vol. 33, pp. 299–312, 2017.

[15] M. Condoluci *et al.*, "Toward 5G densenets: architectural advances for effective machine-type communications over femtocells," *IEEE Commun. Mag.*, vol. 53, pp. 134–141, 2015.

[16] C. Marche, L. Atzori, V. Pilloni, and M. Nitti, "How to exploit the social internet of things: Query generation model and device profiles' dataset," *Comput. Netw.*, vol. 174, p. 107248, 2020.

[17] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proc. PMLR ICML*, 2020, pp. 4387–4398.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. PMLR AISTATS*, 2017, pp. 1273–1282.

[19] Q. Meng, W. Chen, Y. Wang, Z.-M. Ma, and T.-Y. Liu, "Convergence analysis of distributed stochastic gradient descent with shuffling," *Neurocomputing*, vol. 337, pp. 46–57, 2019.

[20] J. Konečný *et al.*, "Federated learning: Strategies for improving communication efficiency," in *Proc. NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[21] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource constrained distributed machine learning," in *Proc. IEEE INFOCOM*, 2018, pp. 63–71.

[22] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas in Commun.*, vol. 37, pp. 1205–1221, 2019.

[23] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*. IEEE, 2019, pp. 1387–1395.

[24] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE ICC*, 2019, pp. 1–7.

[25] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, 1986.

[26] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. NeurIPS*, 2018, pp. 7663–7673.

[27] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proc. PMLR ICLR*, 2020.

[28] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *Proc. PMLR ICML*, 2020, pp. 5381–5393.

[29] P. Rost *et al.*, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 84–91, 2016.

[30] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2017.

[31] P. Schulz *et al.*, "Network architectures for demanding 5G performance requirements: Tailored toward specific needs of efficiency and flexibility," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 33–43, 2019.

[32] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM*, 2018, pp. 207–215.

[33] Y. Xiao and M. Krunz, "QoE and power efficiency trade-off for fog computing networks with fog node cooperation," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.

[34] S. Wang *et al.*, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, 2017.

[35] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE ICDCS*, 2017, pp. 1281–1290.

[36] C. Zhang, H. Du, Q. Ye, C. Liu, and H. Yuan, "DMRA: A decentralized resource allocation scheme for multi-sp mobile edge computing," in *Proc. IEEE ICDCS*, 2019, pp. 390–398.

[37] M. K. Pedhadiya, R. K. Jha, and H. G. Bhatt, "Device to device communication: A survey," *J. Netw. Comput. Appl.*, vol. 129, pp. 71–89, 2019.

[38] W. Sun, E. G. Ström, F. Brännström, Y. Sui, and K. C. Sou, "D2D-based V2V communications with latency and reliability constraints," in *Proc. IEEE GLOBECOM Workshops*, 2014, pp. 1002–1016.

[39] W. Sun, E. G. Ström, F. Brännström, K. C. Sou, and Y. Sui, "Radio resource management for D2D-based V2V communication," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6636–6650, 2015.

[40] Y. Gu, L. X. Cai, M. Pan, L. Song, and Z. Han, "Exploiting the stable fixture matching game for content sharing in d2d-based lte-v2x communications," in *Proc. IEEE GLOBECOM*, 2016, pp. 1–6.

[41] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the internet of things," *IEEE Sys. J.*, vol. 10, no. 3, pp. 1172–1182, 2014.

[42] S. Wang *et al.*, "Opportunistic routing in intermittently connected mobile P2P networks," *IEEE J. Sel. Areas in Commun.*, vol. 31, no. 9, pp. 369–378, 2013.

[43] S.-R. Yang, C.-J. Shih, and P. Lin, "Multi-access edge computing-assisted D2D streaming for proximity-based social networking," in *Proc. IEEE GLOBECOM*, 2019, pp. 1–6.

[44] A. Urpi, M. Bonuccelli, and S. Giordano, "Modelling cooperation in mobile ad hoc networks: a formal description of selfishness," in *Proc. WiOpt*, 2003, pp. 10–pages.

[45] W. H. Yuen, R. D. Yates, and S.-C. Mau, "Exploiting data diversity and multiuser diversity in noncooperative mobile infostation networks," in *Proc. IEEE INFOCOM*, vol. 3, 2003, pp. 2218–2228.

[46] T. Qiu, N. Chen, K. Li, D. Qiao, and Z. Fu, "Heterogeneous ad hoc networks: Architectures, advances and challenges," *Ad Hoc Netw.*, vol. 55, pp. 143–152, 2017.

[47] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. ACM MobiCom*, 2000, pp. 255–265.

[48] X. Li, R. Lu, X. Liang, and X. Shen, "Side channel monitoring: Packet drop attack detection in wireless ad hoc networks," in *Proc. IEEE ICC*, 2011, pp. 1–5.

[49] J. Ko, J. Seo, E.-J. Kim, and T. Shon, "Monitoring agent for detecting malicious packet drops for wireless sensor networks in the microgrid and grid-enabled vehicles," *Int. J. Adv. Robot. Syst.*, vol. 9, no. 1, p. 31, 2012.

[50] K. Edemacu, M. Euku, and R. Ssekibuule, "Packet drop attack detection techniques in wireless ad hoc networks: A review," *Int. J. Netw. Secur. Appl.*, vol. 6, no. 5, p. 75, 2014.

[51] A. E. Clementi, P. Penna, and R. Silvestri, "On the power assignment problem in radio networks," *Mob. Netw. Appl.*, vol. 9, no. 2, pp. 125–140, 2004.

[52] J. Beauquier, J. Burman, F. Dufoulon, and S. Kutten, "Fast beeping protocols for deterministic MIS and $(\Delta+1)$-coloring in sparse graphs," in *Proc. IEEE INFOCOM*, 2018, pp. 1754–1762.

[53] H. Lakhlef, M. Raynal, and F. Taïani, "Vertex coloring with communication constraints in synchronous broadcast networks," *IEEE Trans Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1672–1686, 2018.

[54] I. Jemili, D. Ghrab, A. Belghith, B. Derbel, and A. Dhraief, "Collision aware coloring algorithm for wireless sensor networks," in *Proc. IEEE IWCMC*, 2013.

[55] K. Drira, H. Seba, B. Effantin, and H. Kheddouci, "Distance edge coloring and collision-free communication in wireless sensor networks," *Networks*, vol. 62, no. 1, pp. 35–47, 2013.

[56] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Am. Math. Soc.*, vol. 7, no. 1, pp. 48–50, 1956.

[57] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[58] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," *J. ACM*, vol. 68, no. 1, Jan. 2021.

[59] A. Abdallah, M. M. Mansour, and A. Chehab, "Power control and channel allocation for D2D underlaid cellular networks," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 3217–3234, 2018.

[60] P. Toulis and E. M. Airoldi, "Asymptotic and finite-sample properties of estimators based on stochastic gradients," *Annals of Statistics*, vol. 45, no. 4, pp. 1694–1727, 2017.

[61] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proc. of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

Jian-Jhih Kuo (S'13-M'16) received the Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 2014. He is currently an assistant professor in the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. He was a postdoctoral fellow in the Institute of Information Science, Academia Sinica, Taiwan. His research interests include mobile edge computing, distributed computing, and software-defined networking. He is now a member of IEEE.

Cheng-Wei Ching (S'20) received the BS degree from the Department of Foreign Language, Tamkang University, Taiwan, in 2015. He is currently working toward the MS degree in the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. His research interests include approximation algorithm, mobile edge computing, and machine learning. He is now a student member of IEEE.

Hung-Sheng Huang (S'21) received the BS degree from the Department of Computer Science and Information Engineering, Fu Jen Catholic University, Xinzhuang, Taiwan, in 2019. He is currently working toward the MS degree in the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. His research interests include distributed computing, edge computing, and machine learning. He is now a student member of IEEE.

Yu-Chun Liu received the BS degree from the Department of of Electrical Engineering, National Chung Cheng University, Taiwan, in 2021. He is currently working toward the MS degree in the Institute of Data Science and Engineering, National Yang Ming Chiao Tung University, Taiwan. His research interests include approximation algorithm, distributed computing, and machine learning.