Deep Learning-Based Scheduling Scheme for IEEE 802.15.4e TSCH Network

Md. Niaz Morshedul Haque, Young-Doo Lee, and Insoo Koo

Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, Republic of Korea

Correspondence should be addressed to Insoo Koo; iskoo@ulsan.ac.kr

Received 23 December 2021; Revised 9 February 2022; Accepted 19 February 2022; Published 18 March 2022

IEEE 802.15.4e time-slotted channel hopping (TSCH) is one of the most reliable resources of the Industrial Internet of Things (IIoT). TSCH operates on the slot-frame structure consisting of multiple channel-offsets and multiple slot-offsets. It is gaining acceptance due to its simple architecture and consume low power in industrial applications. The performance of TSCH is mainly dominated by the media access control (MAC) mechanism, which covers the refitment, enumeration, composition, and data transmission. However, in many cases, the data transmission schedules are not accurately prescribed. Therefore, most researchers are trying to define many pragmatic scenarios of scheduling. Their fundamental approach is to schedule TSCH network in a centralized way while framing scheduling based on network performance such as throughput and delay. In this work, a deep learning (DL)-based scheme has been proposed. TSCH network schedules for links to cell assignment of a slot-frame can be constructed as a maximum edge weighted bipartite matching approach. In this paper, we design bipartite edge weight to be composed of throughput and delay, and we use the Hungarian algorithm for proper cell assignment. With the Hungarian scheduling algorithm, we generate the training data and train a deep neural network (DNN) accordingly. In the simulation, we consider a simple TSCH network with 5 nodes where 12 links are formulated, and we consider 16 cells for the link assignment. The simulation results show that the proposed deep learning-based scheduling scheme can provide performance similar to the Hungarian algorithm-based scheduling scheme with above 90% accuracy and nearly 80% execution time reduction.

1. Introduction

The Internet of Things (IoT) is gradually increasing in popularity due to its multifunctionality and handy effectiveness [1]. The Industrial Internet of Things (IIoT) is a promising application of the Internet of Things (IoT). Many industrial appliances can connect through the internet to perform necessary tasks such as real-time observation, industrial automation, security monitoring, and distribution process control [2]. In 2012, the IEEE 802.15.4e standard was announced by IEEE authorities [3]. The IEEE 802.15.4-2015 is the third revision of the IEEE 802.15.4 standard, which is applicable for low-rate wireless networks. Furthermore, the time-slotted channel hopping (TSCH) is one of the operating modes defined in this standard, which is an improved version of the IEEE 802.15.4e [4]. The TSCH follows the media access control (MAC) functionality, which is helpful for industrial applications and operations [5]. Moreover, the TSCH combines multichannel time slot schedule access (in units of slot-frames or superframes) and a channel-hopping mechanism to ensure low power consumption and high reliability [6–8].

The scheduling algorithm is an inevitable aspect of the IEEE 802.15.4e TSCH network. It allocates links to the cells that are a fundamental resource for data transmission. It can either be centralized or distributed; however, it should be established based on the application specific requirements. Here, nodes of the specific network follow a scheduling method that clarifies what is happening in every slot, such as transmit, receive, or remain idle [9]. The slot-frame is the central communication unit for TSCH, which needs a pair of nodes that exchange data. The slot-frame consists
of a set of time slots that repeats continuously over a certain period. The slot-frame used in the TSCH protocol maintains synchronization with network connections. Diverse channels are allocated pseudo-randomly in each time slot, and the scheduling algorithm determines which nearby node to connect with and which channel offset is to be used [10]. In practical applications, TSCH-based scheduling specifies the frequency and slot for every link of a node. We currently see that the TSCH protocol offers a flexible resource allocation considering the channel status depending on traffic nature and surrounding environment [11]. The scheduling algorithms depend upon noncausal information about instantaneous channel qualities, such as the signal-to-noise ratio (SNR) [12]. Furthermore, the TSCH-based scheduling is performed based on the maximization of the network’s throughput [13, 14]. In addition, some other scheduling algorithms utilize previous statistical information on link qualities, such as the expected number of transmissions (ETX) or the packet error rate (PER), to improve the average packet delivery ratio (PDR) [15, 16]. In a real scenario for wireless communication networks, channel state information (CSI) is impacted by several factors of which some are deterministic, and some are random in nature, such as noise in the environment, signal dissipation, channel gain, fading phenomena, and power loss ratio for the interval between transmitting and receiving [17, 18].

This paper proposes a TSCH-based scheduling scheme, which is executed by computing a bipartite graph. The vertexes of the upper side are considered subgroups of noninterfering links, and the lower vertexes are considered as the cells of slot-frame matrix. The edge weight is computed by summating normalized throughput and normalized delay to ensure fairness (details will be discussed in Section 3) [19]. The throughput provides the maximum data transfer, and delay is considered to ensure the reliability. The Hungarian algorithm performs scheduling (cell assignment for links) based on the maximization of bipartite edge weight [20].

In the last couple of years, in several fields of computing, deep machine learning has arisen. With the advances in algorithms for big data and optimization techniques and with more significant computing resource opportunities, deep networks are used as the state-of-the-art approach for numerous issues [21]. Therefore, deep learning (DL) has become one of the vital research routes. It has already played an essential role in machine translation, human voice recognition, image processing and recognition, natural language processing (NLP), computer vision (CV), medical image analysis, and online games. Furthermore, scientists and researchers actively seek to expand these latest technologies in distinct field of wireless communication [22–24].

DL is also used to achieve improved performance over traditional methods in diverse current work in the wireless communication context [25–27]. Several numerical optimization solutions to solve signal processing tasks have already been suggested by distinguish scholars [28–30]. Besides, we have accessed the abundant knowledge of experts in the growth of wireless communications over the past couple of years to complement the data-driven methods and to improve data efficiency by using DL [31–35].

Based on the discussion above, we intend to incorporate the DL method with TSCH-based scheduling and utilize the benefits of DL method to reduce the execution time of scheduling. The proposed deep neural network (DNN) scheme predicts the TSCH-based scheduling, where necessary data required for scheduling are obtained from the Hungarian-based scheduling scheme.

1.1. Literature Review. In this section, we will briefly review the different schemes of TSCH-based scheduling and discuss some contributions of learning-based algorithms to the wireless network’s applications and management. Through this study, we will get an idea about TSCH-based scheduling and its channel state. Furthermore, we will also find a way to propose a learning-based algorithm for TSCH-based scheduling and utilize the advantages of DL.

1.1.1. TSCH-Based Scheduling of Wireless Networks. In the TSCH network, the scheduler controls data transmission opportunities and defines the network features. In the paper, idea and motivation regarding scheduler and their functionalities and properties were taken from the survey [36]. Previously, in TSCH-based scheduling, the influence of link qualities or channel variations was not considered in several cases [37–42]. In these schemes, the authors assumed that the channel status is constant, which means there is no effect of CSI. They solved the scheduling problem by using some tractability of mathematics. However, in real applications of wireless networks, the channels exhibit random behaviours with time-varying conditions. In some papers, we have found that the authors considered the variation of channel status, which means that the CSI was considered for performing the scheduling tasks [9–14].

Specifically, in [9], the authors described a graph and matching theory-based method to maximize throughput and minimize the delay. In addition, they used the Hungarian assignment algorithm [43] to solve the scheduling problem and achieved optimal throughput with channel variations. In [10], the authors presented two TSCH-based scheduling schemes for IEEE 802.15.4e. The schemes were proposed based on statistical CSI (first scheme) and CSI-free scheduling (second scheme). In the first scheme, they considered the network throughput as the edge weight of the bipartite graph. Then, the Hungarian algorithm was used to determine assignment based on maximizing the edge weight of bipartite graph. That means the scheme is able to maximize the throughput of the network. The second scheme proposed a learning-based method, which modelled the problem based on the combinatorial multiarmed bandit (CMAB) approach. This scheme proposed an LLR-based solution of scheduling. Furthermore, this scheme does not require the channel’s information; it was performed based on the feedback knowledge from the previous decision of slot-frame repetition. Finally, both schemes were exhibited the acceptable throughput based on the network capacity. In [11], the authors introduced a dynamic scheduling scheme called the traffic-aware elastic slot-frame adjustment (TESLA) for TSCH-based scheduling. Each node in TESLA adjusted its schedule based on traffic awareness to maximize
energy efficiency while maintaining reliability. In [12], the TSCH-based scheduling scheme was proposed to boost energy efficiency by using Vogel’s approximation method. In this method, the channels chose the nodes based on the highest amount of remaining energy. As a result, the proposed scheme provided higher energy efficiency than many of the previously proposed schemes. In [13], the authors proposed a genetic algorithm (GA)-based suboptimal scheduler to address the scheduling problem by maximizing the throughput while minimizing deadline violations. The proposed solution exhibited better performance than the round-robin and random scheduling schedulers. In [14], the authors proposed a polynomial-time-based scheduling algorithm that maximizes throughput with fairness.

In [37], the authors reduced the slot-frame size by splitting it into units; they called this scheme is “Wave.” The main goal of Wave was to propose a TSCH-based scheduling method by minimizing slots with considering the maximization of throughput and the minimization of delay. The authors introduced orchestra in [38], a scheduling scheme by exploiting the robustness of TSCH to consider a non-deterministic network. Each node was scheduled based on routing protocol (RPL) specifications. In this scheme, there was neither central control nor a signalling node. Though the scheme was considered a simple architecture of scheduling method, nodes were anonymously calculated without considering signalling overhead. Due to this reason, the orchestra was not applicable in real scenarios as different nodes were required different bandwidth with their surroundings. In [39], the authors presented a distributed method for TSCH-based scheduling. They named this scheme “DeAMON.” The main objective of DeAMON was to build a scheduler based on the changeability of topology and reduce the signalling overhead. The DeAMON had performed the scheduling only by considering upward traffic without considering downward traffic. In [40], the authors proposed adaptive static scheduling (ASS) technique to improve the energy efficiency of TSCH network. The ASS extended on static schedules by allowing each pair of communication nodes to adaptively activate a subset of their assigned slots, significantly minimizing unused slots. In [41], the authors presented a centralized adaptive multihop scheduling system (AMUS). The AMUS was established based on traffic awareness and focused on real-time industrial applications. This scheme provided more resources to weak links associated with relay nodes near the sink to reduce idle listening. In [42], the authors presented a traffic-aware scheduling algorithm (TASA), a centralized scheduling technique. The main objective of the TASA was to reduce the delay of TSCH networks. The TASA algorithm used a central node as a coordinator, which must be aware of the entire network topology and the traffic load that each node generated in each slot.

1.1.2. Learning-Based Algorithms for Wireless Networks. Numerous recent reforms have concentrated on learning-based algorithms to reap significant potential benefits in wireless network communication. We got the motivation from research in [44], where the authors provided an outline for applying DL algorithms to the allocation of wireless resources. Here, the authors addressed the constraints of conventional optimization approaches and the scope of DL paradigms in wireless network. DL in radio communication for cloud computing, such as designing a training system for end-to-end wireless communication, demonstrated outstanding performance. However, the low productivity from training time in 5G wireless network and communication system is a constraint when implementing wireless system neural network [45]. DL approaches are not yet widely addressed complicated problems in wireless communication. Still, it is regarded as a critical force and a prominent research topic in several prospective application domains, such as channel estimation, wireless data analysis, mobility analysis, complicated decision-making, service management, and quality enhancement [44, 45]. Deep learning assisted communication network with complex operating condition by accelerating massive amount of computation with assured outcomes. Besides, the authors identified some difficulties and research directions in this critical technology, including a solid mathematical structure, a moderate data set for training, and the need for additional mathematical support to interpret case studies [46]. The authors had projected several DNN methods for different network architectures with varying communication principles, such as the fully connected network and the multilayer feed-forward neural network. There are three layers: an input layer, a hidden layer, and an output layer. Furthermore, all neurons in the previous layer are connected to each neuron in the last layer [47].

Specifically, in [34, 48], the authors incorporated a DL-based DNN scheme to wireless powered communication network (WPCN). The DNN learned the nonlinear mapping and found the optimal solution of different parameters, such as slot allocation, transmit power [34], and energy beam-forming vectors in multiple antennas [48] for WPCN. In both cases, the training data were generated from the iterative sequential parametric convex approximation (SPCA) algorithm. The DNN scheme was trained by SPCA-based training data. After finishing offline training, the DNN showed a similar performance as SPCA and provided a contribution to reduce the computational time. A DL-based scheme was used to estimate channel parameters for wireless resource transfer [49]. Based on input from the energy receivers, the channel parameters were set autonomously at the energy transmitter. The authors have exploited the deep learning ability to optimize the effective throughput of wireless network [50]. They had shown that their extensive experiments were much swifter than the traditional systematic search method indicated in prior studies. Compared to other conventional optimization systems, the convolutional neural network (CNN) based DNN scheme attained high spectral efficiency [51] and less computational time when managing interference.

The learning-based algorithm is still rare in TSCH-based scheduling of wireless network. However, we have found three examples [10, 52, 53] of a learning-based reinforcement learning algorithm for TSCH-based scheduling. In the second scheme of [10], the authors proposed a combinatorial multiarmed bandit (CMAB) approach alongside a
Hungarian-based algorithm to predict the scheduling. In [52], the authors proposed a reinforcement learning-based multiarmed bandit algorithm for selecting the best channel to transmit data packet based on link qualities. As a result, the packet loss was reduced, and better performance was obtained, compared to the switching mechanism. In [53], the authors proposed a multiagent RL-based TSCH scheduling scheme that supports contention to minimize collisions. The TSCH scheduler uses Q-learning to find the most optimum transmission slots. The performance of the proposed TSCH scheduler was consistently better for diverse applications.

1.2. Contributions. The main contribution of this work is to propose a DL-based scheme for TSCH-based scheduling so as to find a fast and more exact solution for the cell assignment of network links. In the previous section, we briefly described many scheduling schemes of TSCH networks; the authors proposed their methods without considering the impact of channel variations [37–42]. These methods assumed that the channel status was constant. Furthermore, the authors used some mathematical tractability to solve the scheduling problem, which is not applicable in pragmatic scenarios of wireless communications in random and time-varying channel conditions. We found some scheduling methods of TSCH network, where the availability of CSI was considered [9–12]. For example, in [9, 10], the proposed scheduling schemes considered the CSI availability for scheduling of TSCH networks.

In this paper, we also assume the CSI availability as random channel status close to the real nature of wireless network. Therefore, the schedule can be changed according to the repetition of the slot-frame. As the channel status changes, the schedule will be performed based on the maximization of bipartite edge weight correlated with CSI. The contributions of the paper can be summarized as follows:

(i) Firstly, we model a bipartite graph for establishing TSCH-based scheduling of IEEE 802.15.4e TSCH networks while considering throughput and delay. The upper vertexes of the bipartite graph constitute all subgroups of noninterfering links and the slot-frame matrix cells as the lower vertexes (details in Section 3). The baseline scheme [10] considered only throughput for a TSCH network. Subsequently, considering the delay in scheduling is our additional contribution compared with the previous method. Specifically, in this paper, the weight of the bipartite graph is computed by considering network throughput and delay. We also focus on ensuring fairness to the optimal use of network resources. For ensuring fairness, we incorporate a window concept to determine the moving average throughput and moving average delay of the network. The throughput and delay are multiplied by the corresponding moving average throughput and delay values to ensure fairness. The moving average throughput and delay are calculated based on the feedback from the previous slot-frame. After determining all parameters, we utilize a prominent Hungarian assignment algorithm [43] to define link to cell assignment based on the maximization of bipartite edge weight (details in Section 3.2). The moving average delay has also ensured the delay between two slot-frames that satisfy the nature of TSCH.

(ii) Secondly, we propose a deep learning-based DNN scheme for TSCH-based scheduling of IEEE 802.15.4e to utilize the advantages of DL. By using the Hungarian algorithm-based scheduling scheme, we generate enough sample data to train the deep learning-based DNN scheme. Then, the supervised offline training of DNN is performed based on the Hungarian algorithm-based scheduling scheme’s input and output. After finishing the training of DNN, in the test phase, the proposed DNN scheme accepts the weight of the bipartite edge as input and offers cell assignments. Thus, the proposed DNN scheme provides a similar performance by learning the relationships between the Hungarian-based scheduling scheme’s input and output while reducing the execution time of scheduling.

(iii) Lastly, the performance of the proposed DL-based DNN scheme is evaluated by simulations. The simulation results show that the proposed DNN scheme can provide similar outcomes to the conventional Hungarian-based scheduling scheme with low execution time.

To the best of our knowledge, this is the first work that addresses scheduling in IEEE 802.15.4e TSCH networks by presenting a deep learning-based scheme. Our proposed DL-based DNN scheme emulates the conventional scheduling method, providing an extra contribution to reduce the execution time of scheduling.

The remainder of the paper is arranged as follows. In Section 2, we demonstrate the system model and different parameters of our proposed model. In Section 3, we describe the proposed Hungarian algorithm-based scheduling scheme and objective function. In Section 4, we delineate the deep learning-based scheme, the architecture of the DL scheme, and the propagation of training and testing data. In Section 5, the proposed method’s outcomes are illustrated and verified by using simulation. Finally, Section 6 concludes the paper.

2. System Model

In this section, we present a system model, which consists of scheduling of TSCH, a TSCH network model, a channel model, and a collision graph. Here, we will discuss the theoretical concepts and introduce a mathematical explanation of the contents.

2.1. Scheduling of TSCH. Scheduling of time-slotted channel hopping in wireless network means specifying links for nodes to transmit data, which allows for an efficient distribution of wireless connections to enhance communications.
The time slot and channel where each node should deliver or receive information from nearby nodes are defined by the scheduling method. Furthermore, the scheduling method is significantly influenced by network performance such as throughput, delay, and durability. The cell is considered as the key component of scheduling. According to the IEEE 802.15.4e guidelines, the duration of each cell is usually 15 ms [9]. The transmitter sends the data packet, and the receiver returns the acknowledgment after successful reception.

It should be mentioned that by combining the channel offset and the absolute slot number (ASN), a node pseudorandomly switches the channel in each time slot. In general, as expressed below, frequency \( f \) can be extracted as follows:

\[
f = \mathcal{F}[\text{ASN} + \text{Ch}_{\text{offset}}]\%
k_h,
\]

where \( \text{Ch}_{\text{offset}} \) represents the channel offset, \( q_h \) is the number of available channels, and \( \mathcal{F} \) denotes the mapping function.

### 2.2. TSCH Network Model

A TSCH network consists of a single gateway access point and multiple nodes, and a gateway connects the nodes to synchronize each node on the single gateway access point and multiple nodes, and a gateway connection exists between them. Every cell \( c \) link \( \text{fi} \) indicated by a radio while providing a contact range \( R_i \). The time slot and channel where each node should deliver or receive information from nearby nodes are defined by the scheduling method. According to the IEEE 802.15.4e guidelines, the duration of each cell is usually 15 ms [9]. The transmitter sends the data packet, and the receiver returns the acknowledgment after successful reception.

The time slot and channel where each node should deliver or receive information from nearby nodes are defined by the scheduling method. Furthermore, the scheduling method is significantly influenced by network performance such as throughput, delay, and durability. The cell is considered as the key component of scheduling. According to the IEEE 802.15.4e guidelines, the duration of each cell is usually 15 ms [9]. The transmitter sends the data packet, and the receiver returns the acknowledgment after successful reception.

It should be mentioned that by combining the channel offset and the absolute slot number (ASN), a node pseudorandomly switches the channel in each time slot. In general, as expressed below, frequency \( f \) can be extracted as follows:

\[
f = \mathcal{F}[\text{ASN} + \text{Ch}_{\text{offset}}]\%
k_h,
\]

where \( \text{Ch}_{\text{offset}} \) represents the channel offset, \( q_h \) is the number of available channels, and \( \mathcal{F} \) denotes the mapping function.

### 2.2. TSCH Network Model

A TSCH network consists of a single gateway access point and multiple nodes, and a gateway connects the nodes to synchronize each node on the network. In a centralized manner, scheduling is carried out where the scheduler is installed in the gateway and specifies the allocation of time and the frequency for transmission of each node.

The TSCH network is designed as \( \mathcal{G} = (V, M, d) \) where \( V \) is the set of nodes \( V = \{v_1, \cdots, v_T\} \) and \( M \) is the total number of nodes, \( M \) is the set of links, and \( d \) is the set of physical distances between each pair of adjacent nodes in set \( V \). An accurate distance between node \( A \) and node \( B \) in the TSCH network is denoted by \( d_{A,B} \). Each node \( v_i \) is configured by a radio while providing a contact radius \( \mathcal{R}_i \), greater than the interference range \( \mathcal{R}_i \). The signs \( \ell \in \{1, \cdots, T\} \) and \( f \in \{1, \cdots, F\} \) indicate the time of each slot and the range of network frequencies, respectively.

An example of a slot-channel matrix of a simple TSCH network for a five-node structure of graph \( \mathcal{G} \) is illustrated (see Figure 1). Here, cells are accessed by two noninterference links. In a slot-frame, links such as \( C \rightarrow E \) and \( B \rightarrow A \) both are shared the same cell; here, the position is (Slot offset, Channel offset) = (4, 4). This is possible due to the noninterfering connection between them. Every connection is a computational method that appears in the slot-frame cells.

### 2.3. Channel Model

A combination of channel frequencies \( f \in \{1, \cdots, F\} \) and time of the slot \( \ell \in \{1, \cdots, T\} \) constitutes a cell \( c \in C(= \{1, \cdots, q_C\}) \), and \( q_C \) indicates the number of cells. The state of each channel for specific cell \( c \) assigned to specific link \( m \in M(= \{1, \cdots, q_M\}) \) at slot-frame \( n \) is defined by \( x_{m,c}(n) = |H_{m,c}(n)|^2 \), where \( q_M \) is the number of links and \( H_{m,c}(n) \) indicates the channel gain of cell \( c \) and link \( m \) at slot-frame \( n \).

Rayleigh fading is a feasible model when there are numerous things in the surrounding area that scatter the radio signal prior to it reaching the receiver. In this paper, we assume that the channel gain \( H_{m,c}(n) \) will follow Rayleigh fading, and it will be determined by the following probability density function (pdf):

\[
F_B(r) = \frac{2r}{\Omega} e^{-r/\Omega},
\]

where \( \Omega = E[R^2] \) and \( R \) is a random variable.

When channel gain \( H_{m,c}(n) \) and transmission power \( p \) are given, the number of packets that can be delivered over link \( m \) and cell \( c \) in frame \( n \) is calculated, which is considered as the throughput of network. Further, it can be computed based on Shannon’s formula such that we have:

\[
\theta_{m,c}(x_{m,c}(n)) = \frac{\beta}{T} \exp \left( \frac{q_C - c}{q_T} \right),
\]

where \( N \) indicates the normalization value of delay performance, \( q_T \) is the number of slots, and \( k_0 \) is a constant parameter for evenly adjusting the unit ratio between throughput and delay.

### 2.4. Collision Graph

A collision graph \( Q = (M, C) \) is defined to consider the interference in the proposed network model, where \( C \) represents the links in the collision graph. It’s vertexes are referred to as the configuration graph \( \mathcal{G} \), and its edges reveal the conflict between the two links. It demonstrates the various ways to transmit data during a collision (see Figure 2). However, the transmission such as \( A \rightarrow B \) and \( A \rightarrow C \) do not occur concurrently, and the collision graph has an edge between them. Furthermore, a legitimate schedule is not allowed to send such data through a common cell. In addition, in the collision graph, a scheduling method can be chosen, such as an autonomous set of vertexes to be allocated in a similar cell. Note that in the collision graph, there are no edge between such a pair of vertexes. The purpose of scheduling is that two colliding links are not allocated in the same cell for transmitting data.

In this structure, which uses a slot-frame cell, the links within a specific and clear separation in the collision graph are designed collectively. This technique is performed in a preprocessing stage before executing the scheduling algorithm.
The set $\mathcal{M}$ of links corresponds to the edge weight of the bipartite graph, in this paper, bipartite graph, and slot-frame cells are also considered the bottom vertexes of the bipartite graph. Thus, the edges correspond to the collision graph is considered the top vertexes of the graph edge weight is computed based on the network throughput and delay. The vertexes are separated into two dissociated sets (top and bottom) (see Figure 3). Furthermore, each subset of the noninterference links obtained from the collision graph is considered the top vertexes of the bipartite graph, and slot-frame cells are also considered the bottom vertexes of the bipartite graph. Thus, the edges connect a vertex from one set to a vertex from another set. A proper scheduling is a method in which one vertex of the bottom range of the graph corresponds to any top vertex of the bipartite graph based on the maximization of bipartite edge weight.

Assume $B = (M, \mathcal{C}, E)$ is a bipartite graph (see Figure 3). The set of links $M = \{1, \cdots, q_M\}$ corresponds to the collision graph $Q$, which is modelled as the top vertexes of the bipartite graph. The set of cells $\mathcal{C} = \{1, \cdots, q_c\}$ corresponds to the slot-frame matrix cells, $\mathcal{S}$, which is also considered the bottom vertexes of the bipartite graph. The set $E = \{(e = (m, c) | m \in M, c \in \mathcal{C})\}$ is the edges of bipartite graph $\mathcal{B}$. The weight, $W_{mc}(n)$, of edge $e = (m, c)$ at slot-frame $n$ is calculated as follows:

$$W_{mc}(n) = \alpha u_m^{N}\psi_{mc}^{N}(n) + (1 - \alpha)u_m^{N}\psi_{mc}^{N}$$

where $\alpha$ is the weighting factor between throughput and delay. In this paper, we calculate throughput by multiplying normalized throughput $\theta_{mc}^{N}(n)$ by the normalized moving average for throughput $u_m^{N}$. Similarly, we calculate the delay by multiplying delay performance $\psi_{mc}^{N}$ by the normalized moving average for delay $u_m^{N}$. We consider this approach to ensure fairness in bipartite edge weights.

Normalized throughput $\theta_{mc}^{N}(n)$, for cell $c$ and link $m$ in slot-frame $n$, can be achieved from the ratio of each specific value to the maximum value of the corresponding slot-frame by using the following equation:

$$\theta_{mc}^{N}(n) = \frac{\theta_{mc}(x_{mc}(n))}{\max_{j \in \mathcal{C}} \theta_{ij}(x_{ij}(n))}$$

The moving average throughput $u_m^0$ and the moving average delay $u_m^1$ are used in a calculation to analyze data points by generating a sequence of averages for a finite-value set of the various subsets of the complete data set. Only the recent values for throughput and delay use as moving-average approaches with consideration for window size $k_w$. The moving average throughput and the moving average delay are determined based on the link assignment of slot-frame of the TSCH network. Therefore, the link-wise moving average throughput $u_m^0$ and the moving average delay $u_m^1$ are calculated by the following equations:

$$u_m^0 = u_m^0(n - 1) + \frac{\theta_{mc}(n - 1) - \theta_{mc}(n - k_w - 1)}{k_w}$$

$$u_m^1 = u_m^1(n - 1) + \frac{\psi_{mc}(n - 1) - \psi_{mc}(n - k_w - 1)}{k_w}$$

For a specific slot-frame $n$, if a link $m$ assigned to a cell $c^*$, then the throughput for this specific assignment is denoted as $\theta_{mc^*}(n)$. Similarly, the delay performance for this specific slot-frame $n$ for this specific assignment (link $m$ assigned to a cell $c^*$) is denoted as $\psi_{mc^*}(n)$. The $\theta_{mc^*}(n)$ and $\psi_{mc^*}(n)$ are determined for all links $m \in M$. Since we assumed the
availability of CSI into our proposed model, that is why the assigned cell $c^*$ will change for the specific link $m$ over the repetition of the slot-frame $n$.

Normalized moving averages for throughput $u_m^{N0}$ and delay $u_m$ can be obtained with the following equations:

$$u_m^{N0} = 1 - \frac{\exp{(u_m^0)}}{\sum_{m \in M} \exp(u_m^0)}, \quad (9)$$

$$u_m^{N1} = \frac{\exp{(u_m^1)}}{\sum_{m \in M} \exp(u_m^1)}. \quad (10)$$

3.1. Problem Definition and Objective Function. In this section, we define the maximization of bipartite edge weight as a prologue to our strategy under random channel distribution (CSI availability). Our objective is to assign corresponding links $m \in M$ to cells $c \in C$ of all slot-frames $n \in \mathbb{N}$. This assignment will be performed based on the maximization of bipartite edges $e \in E$ weight $W_{me}(n)$ (defined by equation (5)) correspond to the throughput and delay.

If we characterize $\xi_{n,me}$ as a binary decision variable, the weight maximization process is formulated with the following equation:

$$W_T^* = \max \sum_{n \in \mathbb{N}} \sum_{e \in E} \sum_{m \in M} \sum_{c \in C} \xi_{n,me} W_{me}(n). \quad (11)$$

3.2. The Hungarian Assignment Algorithm. We utilize the prominent Hungarian assignment algorithm, which performs the scheduling task based on the maximization of bipartite graph edge weight to satisfy our objective (defined in equation (11)) [20, 43]. We consider the bipartite edge weight $W_{me}(n)$ (defined equation (5)) as the cost matrix of the Hungarian assignment algorithm, where the total number of edges is the multiplication of the total number of links (the top vertexes of the bipartite graph) and the total number of cells (the bottom vertexes of the bipartite graph) ($|q_M \times q_C|$). Furthermore, the Hungarian algorithm operates in polynomial time. Therefore, we consider the negligible upper bounds $2^L$ ($L$ is the communication path between two nodes of the topology graph) [10].

The Hungarian-based scheduling algorithm is shown in Algorithm 1.

We observe the single slot-frame cell assignment (see Figure 4), where the links (obtained from collision graph with a subset of noninterference links) are assigned by the different cells. It reveals that there is no conflict in the scheduling that satisfies the nature of TSCH. We also observe that the cell scheduling is performed based on the maximization of bipartite edge weight (see Figure 5) allocation for all links.

4. The Proposed Deep Learning-Based Scheduling Scheme

In this section, our aim is to establish a supervised DL-based deep neural network (DNN) scheme, which can achieve an optimal approximation of the Hungarian algorithm-based scheduling scheme of the TSCH network.

4.1. Proposed Algorithm for Generating a Data Set. Algorithm 2 summarizes the proposed Hungarian algorithm-based scheduling scheme and will be used to generate data for DNN training. Algorithm 2 reflects the total system model of our proposed scheduling scheme. Firstly, we must set all constant parameters. We have seen that the moving average throughput and moving average delay are dependent on the immediate last slot-frame parameters. That is why we have also set the initial moving average throughput value and the moving average delay value for the first slot-frame scheduling. After determining the first slot-frame scheduling, the specific parameters will be obtained; the next scheduling will be determined automatically.

From Algorithm 2, specifically, we should focus on moving average delay. Here, we observe that the present moving average delay of slot-frame is dependent on the immediate last slot-frame. That is why the knowledge from the feedback of the previous slot-frame is needed for determining the scheduling every slot-frame. It means the present slot frame must wait for the moving average delay value from the immediate last slot-frame for executing its own scheduling. Now, we can say that the moving average delay is the time between two slot-frames. Therefore, the moving average delay $u_m(n)$ can be considered as a slot-frame delay, which satisfies the nature of TSCH.

4.2. Structure of the DL-Based DNN Scheme. A simple multi-layer perceptron (MLP) is a basic frame of the proposed DL-based DNN scheme. The MLP is a neural network that is fully connected and comprises one input layer, several hidden layers, and one output layer (see Figure 6). We consider the input of DNN as the bipartite edge weight $W_{me}(n)$, which was considered the cost matrix of the Hungarian-based scheduling scheme. Suppose we recall the Hungarian-
based scheduling scheme, where specific cells $c \in \mathbb{C}$ are assigned for all links $m \in M$. However, in that case, the output of DNN is considered as the cell scheduling $c^*$, which is determined by a Hungarian-based scheduling scheme (Algorithm 2).

From the discussion mentioned above, now it is clear that the total number of inputs of DNN is the total number of bipartite edges ($e \in \mathcal{E}$), which can be determined by the multiplication of the number of links $q_M$ and number of cells $q_C$. Therefore, the total number of inputs of DNN

**Algorithm 1: The Hungarian-based scheduling algorithm.**

- **Step 1.** Insert data $\rightarrow$ Input edge weight from bipartite graph ($\langle q_M \ast q_C \rangle$).
- **Step 2.** $a =$ cost matrix % edge weight of bipartite graph.
- **Step 3.** $b =$ max ($a$) % determine maximum value of cost matrix.
- **Step 4.** $y = b - a$ % the process is to subtract all elements from the highest value of the cost matrix.
- **Step 5.** The row operation is executed % subtract row minima from each, row-wise.
- **Step 6.** The column operation is executed % subtract column minima from each, column-wise.
- **Step 7.** Mark the lowest zero element and remove other zero elements.
- **Step 8.** Find the minimum value of all uncovered elements.
- **Step 9.** The minimum value is subtracted from uncovered elements and added to intersect elements.
- **Step 10.** If column and row are uncovered, the predecessor index is updated.
- **Step 11.** The optimal assignment has achieved based on the maximization of bipartite edge weight.

**Figure 4:** Cell assignment by the proposed Hungarian algorithm-based scheme.

**Figure 5:** Maximized the bipartite edge weight allocation in a single slot-frame.
is \( q_M \cdot q_C \). Furthermore, the number of outputs is the total number of links, which is \( q_M \cdot q_C \).

Neurons in hidden layers and the number of hidden layers are hyperparameters in the DNN and need to be modified to attain maximum accuracy in the optimal solution. The following equation can describe the output of each \( h \)th layer in the hidden layers:

\[
e_h = g(e_{h-1}w_h + b_h),
\]

where \( w_h \) and \( b_h \), respectively, are the weights and biases of the \( h \)th layer. The learning is a question of determining the weights \( w \) within a specific possible set that will result in the output that illustrates the best input mapping, and \( g(.) \) implies an activation function that adopts each hidden layer’s output.

A DNN generally executes the sum of input products and their corresponding weights and executes the \( g(.) \) activation function develops for each output of the hidden layers. The activation function explicitly allows the network to acquire nonlinear mapping of input and output.

We use the rectified linear unit (ReLU) activation function in the proposed scheme for the output of each hidden layer. The ReLU is a simple function to implement in the following form:

\[
g(x) = \max(0, x)
\]

Algorithm 2: Hungarian algorithm-based scheduling to generate DNN training data.

```plaintext
1: Initialize: constant parameters \( \beta, l, p, \eta, q_M, q_C, a, k_w \) and \( k_b \).
Make an enforceable level as the initial point for equation (5), for initial moving average throughput \( u_m^0(1) \), and for initial moving average delay, \( u_m^0(1) \) % needed for first slot-frame scheduling
2: //initial loop:
3: For each \( m \in M \) and \( c \in \mathbb{C} \) of \( e \in E \):
4: Run pdf of Rayleigh distribution to determine channel gain \( H_{mc}(n) \) of the network
5: Update the channel state \( x_{mc}(n) \)
6: Accordingly, determine the throughput \( \theta_{mc}(x_{mc}(n)) \) with equation (3) and delay performance \( \psi_{mc}(c) \) with equation (4) % the \( \theta_{mc} \) changes for every slot-frame change, but \( \psi_{mc} \) is fixed for all slot-frames
7: Normalized throughput \( \theta_{mc}(x_{mc}(n)) \) is executed with equation (6)
8: Normalized moving average throughput \( u_m^N \) and delay \( u_m^1 \) are determined by equation (9) and equation (10) respectively % the first frame develops by using the initial value of moving average parameters after obtaining the first schedule, and ensures \( c^*; \) for \( n \rightarrow n + 1 \) these parameters will be updated with equation (7) and equation (8), respectively
9: Build an edge weight \( W_{mc}(n) \) with equation (5)
10: Run Hungarian algorithm in Algorithm 1 to find a link to the cell, matching it to get the maximum weighted matching \( \mathcal{W}_T \) of bipartite graph \( \mathcal{B} = (M, \mathbb{C}, E) \)
11: //main loop:
12: Generate 10,000 data frames for DNN training
```

Figure 6: The proposed fully connected multilayer DNN structure, which comprises one input layer, four hidden layers, and one output layer.
DNN and is computationally efficient and faster [33, 34]. The output of the ReLU activation function can be determined with \( g(r) = \max(0, r) \), such that if \( r < 0, g(r) = 0 \); otherwise, \( g(r) = r \).

4.3 The DNN Training. The essential part of the DL-based scheme is the training of DNN. A perfectly trained DNN learns the correlation between input and output. In supervised learning, we require enough input and output data to train the neural network, from which the DNN recognizes the kinship between input and output. In the proposed DNN scheme, we equip the neural network from training data consisting of samples in an iterative method developed using the Hungarian algorithm-based scheduling scheme as previously discussed. The formation of neural network is a mechanism by which the loss function is reduced to classify network parameter \( w \). Furthermore, the loss function is between the actual scheduling \( c^* \) and the predicted DNN output optimal assignment \( \hat{c}^* \). Thus, the loss function should reduce what we consider the mean squared error (MSE) by the following equation:

\[
J(w) = \mathbb{E} \left( [c^* - \hat{c}^*]^2 \right).
\]

We employed the Adam optimizer to mitigate the MSE loss function in our proposed DNN scheme. The method is simple to execute, is effective in computing, and needs minimum memory.

A total of 10,000 data samples were generated using Algorithm 2 based on the Hungarian algorithm-based scheduling scheme. The training samples were divided into three data segments. To train the DNN, we took 60% of the data while keeping 20% for validation and the other 20% for testing. Validation is an unbiased method of testing the network when training the model. Validation of the DNN model’s reliability on the training data set biases the score outcomes. Therefore, we used 20% of the data set to validate the unbiased assessment. We used this method to assess how well our scheme was learned during training. Testing is an actual, unbiased performance analysis of the DNN. Furthermore, we employed the data normalization technique to prevent training and validation errors due to overfitting.

Figure 7 shows the training process for the proposed DNN scheme. The network environment generates corresponding data frames or sample data. The supervised offline training is performed under this process and will obtain an optimal approximation of the Hungarian-based scheduling algorithm. After finishing the DNN training, we could use it for every test value of the bipartite edge weight, \( W_{mc} \), to trace the optimal solution, \( c^* \).

5. Simulation Results and Performance Evaluation

We compare the outcomes provided by the DNN with the Hungarian algorithm-based scheduling scheme to demonstrate the efficiency of our proposed DL-based DNN scheme. For generating data, we used Algorithm 2, which projected our whole system model. We generated 10,000 sample data for a different value \( \alpha \). After obtaining corresponding data samples, we trained the DNN and found the optimal parameters. The TSCH network simulation for data set generation was carried out in MATLAB on a PC with an Intel Core i7 processor and 8GB RAM. After extracting data samples, the supervised offline training of DNN was executed in Python utilizing the Keras and NumPy libraries.

5.1 TSCH Network: Bipartite Graph Establishment. As described earlier, this work’s fundamental goal is to establish a bipartite graph model of the TSCH network. The model detailed is illustrated in Algorithm 2 to generate enough samples data to train the DNN correctly. Based on our network model, we considered the parameters to establish the described model in Table 1.
5.2. Construction of the DNN Scheme. In this section, we construct a DNN scheme, which can obtain an optimal approximation of the Hungarian algorithm-based scheduling scheme for an IEEE 802.15.4e TSCH network. A DNN with one input layer, multiple hidden layers, and one output layer is considered. The input feature of the DNN is given as the bipartite edge weight $|W_{mc}(n)|$. The number of input features should be the total number of edges in the bipartite graph, which is a multiplication of the number of links, $|q_M|$, and the number of cells, $|q_C|$. In our example, the number of input features is $|q_M| * |q_C| = 12 * 16 = 192$. The output is set as the Hungarian cell scheduling output for all links $m \in M$. A total of 12 links in the example would be assigned to 16 cells, which indicates that the number of outputs from the DNN will be 12. Specifications for the proposed DNN scheme are given in Table 2. When ReLU activation is used in every hidden layer, a minimal MSE is obtained with little difficulty.

We trained the DNN on 6000 data samples for different weighting factors, $\alpha$, and validated the outcomes in all training epochs on 2000 data samples. We conducted tests on 2000 data samples and proved that the proposed DNN scheme achieved an intuitive mechanism for scheduling, as good as the Hungarian algorithm.

5.3. Performance Evaluation

5.3.1. Determining the Model Accuracy. Accuracy is the most vital part of the neural network, and it indicates how much the proposed learning-based scheme can precisely learn the previously proposed scheme. We developed an accuracy metric based on the number of assigned cells between the Hungarian algorithm-based (HG) scheduling scheme and the proposed DL-based DNN scheme. First, we measured the accurate parameters. Based on our proposed model, we measure accurate parameters by using the following method. The output of HG scheduling (assigned cell number) is an integer, but the output of DNN or scheduling prediction is not an integer value. For mitigating this problem, we use the round function to find integer values predicted schedule. We measured the accuracy for 1000 test samples as depicted in Table 3, such that $N_t = 1000$. Accuracy can be determined with equation (14) for different values of weighting factor $\alpha$. The significant of $\alpha$ will be discussed in the next section.

$$\text{Accuracy} = \frac{N_t * q_M - |P|}{N_t * q_M} * 100\%.$$  

5.3.2. Fairness: Bipartite Edge Weight. Figure 8 shows the fairness of our proposed scheduling scheme and that of the baseline scheme [10] in terms of bipartite edge weight. As discussed earlier, bipartite edge weight in the baseline scheme [10] was only computed by considering throughput. One of the contributions of this paper is to ensure fairness on bipartite edge with considering throughput and delay. We utilize window concepts to determine moving average parameters and multiply corresponding normalized throughput and delay parameters to ensure fairness. The bar chart shows that bipartite edge weight in the baseline scheme [10] has 1.36 as the maximum value and 0.7 as the minimum value. On the other hand, the bipartite edge weight of the proposed scheme in the paper has 1.37 as the maximum value and 1.1 as the minimum value. As a result, it is shown that our proposed scheme gives more fairness than the baseline scheme [10].

5.3.3. Throughput and Delay Impact on Bipartite Edge Weight. Figure 9 shows the throughput and delay impact on the bipartite edge weight of 1000 test samples for the Hungarian algorithm-based scheduling scheme (HG) and

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of links, $q_M$</td>
<td>12</td>
</tr>
<tr>
<td>Set of links, $M$</td>
<td>1 : 12</td>
</tr>
<tr>
<td>Number of cells, $q_C$</td>
<td>16</td>
</tr>
<tr>
<td>Set of cells, $C$</td>
<td>1 : 16</td>
</tr>
<tr>
<td>Number of slots, $q_S$</td>
<td>4</td>
</tr>
<tr>
<td>Number of data frames, $N$</td>
<td>10,000</td>
</tr>
<tr>
<td>Bandwidth, $\beta$</td>
<td>1 MHz</td>
</tr>
<tr>
<td>Bits in each packet, $I$</td>
<td>1000</td>
</tr>
<tr>
<td>Transmission power, $p$</td>
<td>10 MW</td>
</tr>
<tr>
<td>Noise variance, $\eta$</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Parameter specifications for Algorithm 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of input features</td>
<td>192</td>
</tr>
<tr>
<td>Number of hidden layers</td>
<td>4</td>
</tr>
<tr>
<td>Number of neurons in each layer</td>
<td>800, 1600, 1200, and 800</td>
</tr>
<tr>
<td>Number of outputs</td>
<td>12</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Batch size</td>
<td>100</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 2: Parameter specifications for the DNN model.

<table>
<thead>
<tr>
<th>Weighting factor</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0.1$</td>
<td>92</td>
</tr>
<tr>
<td>$\alpha = 0.5$</td>
<td>93</td>
</tr>
<tr>
<td>$\alpha = 0.9$</td>
<td>92</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of the proposed DNN scheme.
DL-based DNN scheme. Figure 9(a) shows the normalized average throughput from HG and the DNN according to the weighting factor $\alpha$. Figure 9(b) shows the normalized average delay from HG and the DNN according to the weighting factor $\alpha$. We generated several data sets with different values for $\alpha$ and trained the DNN, where $\alpha$ is the weighting factor between throughput and delay. According to equation (5), $\alpha$ it will affect throughput and delay performance on bipartite edge weight. The impact of the weighting factor $\alpha$ is summarized in Table 4. It shows that when the weighting factor $\alpha$ is closer to zero, the throughput impact on the bipartite edge is lower, and delay performance is higher. As the weighting factor $\alpha$ gets close to 1, throughput impact is more elevated, and subsequently, delay performance is lower.

From Figure 9, it is also observed that the DNN scheme shows a similar performance and correctly emulates the conventional HG-based scheduling scheme. However, due to the prediction and accuracy metric (Section 5.3.1), the DNN is slightly lower than HG.

### 5.3.4. Performance of DNN Accuracy: Example of Cell Scheduling

As a case, we select 20 test samples at random for the performance accuracy verification between Hungarian (HG) and DNN. We observed that on an average of 17 times, our proposed DNN scheme performed similar cell scheduling to the HG scheduling scheme. For the remaining 3 samples, only a few links were differently assigned. Figure 10 shows two examples of the cell scheduling pattern in our proposed scheme. Figure 10(a) shows one of the cases where the cell scheduling between HG and the DNN is similar for all links. Figure 10(b) shows the case where only one link is differently assigned. For HG scheduling, link index 12 is assigned in cell number 11, but for DNN cases, it is assigned in cell number 10, and the rest of the links are assigned accurately for both cases. This scenario supports achieving above 90% accuracy (Section 5.3.1) for different weighting factor values $\alpha$.

### 5.3.5. Execution Time

Finally, the execution time needed for both schemes HG and DNN, based on the number of data samples, is shown in Figure 11. It is observed that the execution time of the DNN scheme is much lower than that of the Hungarian-based scheme. As a result, utilizing the DNN for cell scheduling in IEEE 802.15.4e TSCH networks is computationally efficient compared to the Hungarian algorithm-based scheduling scheme. Our proposed DNN scheme provided similar trends to the previously studied case [34, 48], where power allocation is considered.

### 5.4. Case Studies of Different Scenarios

Different TSCH-based scheduling schemes had been proposed for the case of constant CSI or CSI availability. We have found three learning-based algorithms in TSCH-based scheduling. The contributions of these schemes were to maximize the network throughput, to minimize delay, to improve energy efficiency, to reduce packet loss or to consider changeability in slot-frame size, etc. Therefore, we summarize some case studies according to different scheduling algorithms used to perform TSCH-based scheduling. The various case studies are shown in Table 5.
Figure 9: The impact of throughput and delay on bipartite edge weight: (a) normalized average throughput according to weighting factor $\alpha$; (b) normalized average delay according to weighting factor $\alpha$. 
Table 4: The impact of throughput and delay on bipartite edge weight.

<table>
<thead>
<tr>
<th>Weighting factor</th>
<th>Throughput</th>
<th>Delay performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$ close to 0</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>$\alpha = 0.5$</td>
<td>Balanced</td>
<td>Balanced</td>
</tr>
<tr>
<td>$\alpha$ close to 1</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Figure 10: Examples of cell scheduling for the proposed DNN-based scheme and HG-based scheme: (a) all link indexes in scheduling between the DNN and HG are similar; (b) dissimilarity between the DNN and HG for link index 12.
Figure 11: Execution time according to the number of data samples for the Hungarian scheme and the DNN scheme.

Table 5: Case studies of TSCH-based algorithms.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Scheduling algorithm</th>
<th>CSI availability</th>
<th>Learning-based algorithm</th>
<th>Contributions</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>Wave</td>
<td>No</td>
<td>No</td>
<td>(i) Reduction of the slot-frame size was considered</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ii) The throughput was maximized, and the delay was minimized</td>
<td></td>
</tr>
<tr>
<td>[38]</td>
<td>Orchestra</td>
<td>No</td>
<td>No</td>
<td>(i) RPL specification was proposed</td>
<td>2015</td>
</tr>
<tr>
<td>[39]</td>
<td>DeAMON</td>
<td>No</td>
<td>No</td>
<td>(i) Changeability of topology was considered</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ii) The signalling overhead was reduced</td>
<td></td>
</tr>
<tr>
<td>[40]</td>
<td>ASS</td>
<td>No</td>
<td>No</td>
<td>(i) The energy efficiency was improved</td>
<td>2018</td>
</tr>
<tr>
<td>[41]</td>
<td>AMUS</td>
<td>No</td>
<td>No</td>
<td>(i) Traffic awareness reduced idle listening</td>
<td>2016</td>
</tr>
<tr>
<td>[42]</td>
<td>TASA</td>
<td>No</td>
<td>No</td>
<td>(i) The delay was reduced</td>
<td>2013</td>
</tr>
<tr>
<td>[12]</td>
<td>Optimal greedy method</td>
<td>Yes</td>
<td>No</td>
<td>(i) The energy efficiency was improved</td>
<td>2017</td>
</tr>
<tr>
<td>[13]</td>
<td>GA-based suboptimal scheduler</td>
<td>Yes</td>
<td>No</td>
<td>(i) The throughput was maximized while minimizing the deadline constraints</td>
<td>2017</td>
</tr>
<tr>
<td>[14]</td>
<td>Polynomial-time-based scheduling algorithm</td>
<td>Yes</td>
<td>No</td>
<td>(i) The throughput was maximized with consideration of fairness</td>
<td>2018</td>
</tr>
<tr>
<td>[9]</td>
<td>Graph and matching theory-based strategy: Hungarian assignment algorithm</td>
<td>Yes</td>
<td>No</td>
<td>(i) The throughput was maximized, and the delay was minimized</td>
<td>2016</td>
</tr>
<tr>
<td>[52]</td>
<td>Reinforcement learning</td>
<td>Yes</td>
<td>MAB</td>
<td>(i) The packet loss was reduced</td>
<td>2018</td>
</tr>
<tr>
<td>[53]</td>
<td>Multiagent RL-based TSCH scheduling algorithm</td>
<td>Yes</td>
<td>Q learning</td>
<td>(i) The collisions were minimized, and the packet latency was reduced</td>
<td>2020</td>
</tr>
<tr>
<td>[10]</td>
<td>Bipartite graph-based approach: Hungarian assignment algorithm (scheme-1)</td>
<td>Yes</td>
<td>CMAB</td>
<td>(i) The throughput was maximized</td>
<td>2020</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ii) The CMAB approach was proposed which showed similar performance to Hungarian-based scheduling algorithm without considering CSI</td>
<td></td>
</tr>
<tr>
<td>This paper</td>
<td>Bipartite graph-based approach: Hungarian assignment algorithm</td>
<td>Yes</td>
<td>DNN</td>
<td>(i) The impact of throughput and delay on the TSCH network was considered to ensure fairness</td>
<td>2020</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ii) DNN scheme demonstrated the similar performance with lower execution time to Hungarian-based scheduling scheme</td>
<td></td>
</tr>
</tbody>
</table>
6. Conclusion

In this paper, we proposed a DL-based scheduling scheme for IEEE 802.15.4e TSCH network where a DNN was trained to learn the nonlinear mapping of network parameters, i.e., cell scheduling, throughput, and delay. We executed scheduling based on maximization bipartite edge weight. The data for training were generated by an iterative method based on the Hungarian algorithm-based scheduling scheme. The simulation results showed that the proposed DL-based DNN scheme provided the same accuracy to the traditional iterative Hungarian algorithm-based scheduling scheme while offering lower execution time. In addition, if the traffic load increases, the number of nodes and links will increase. In such cases, the Hungarian-based scheduling scheme will face difficulties in terms of requiring more computational time. Therefore, the proposed DL-based DNN scheme will be more proper in high traffic load since it can perform the cell assignment for every test value of bipartite edge weight in the real fashion of a TSCH-based scheduling scheme. Furthermore, as one of future work, we will consider uplink/downlink characteristics to enhance network performance and implement the proposed scheme in embedded devices.

Data Availability

No public/others data were used to support this study.

Conflicts of Interest

The authors declare they have no conflict of interest.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea through the Korean Government Ministry of Science and ICT (MSIT) under Grant NRF-2021R1A2B5B01001721. In addition, this result was supported by the “Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-003).

References

Deep reinforcement learning for dynamic multichannel access in wireless networks,


S. Duquennoy, Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH Categories and Subject Descriptors337–350.