TSCH-BASED SCHEDULING FOR THROUGHPUT AND DELAY OPTIMIZATION OF IEEE 802.15.4e: A DEEP LEARNING-BASED APPROACH

——————————

DISSERTATION

for the Degree of

MASTER OF SCIENCE
(Electrical Engineering)

——————————

MD. NIAZ MORSHEDUL HAQUE

OCTOBER 2021

TSCH-Based Scheduling for Throughput and Delay Optimization of IEEE 802.15.4e: A Deep Learning-Based Approach

DISSERTATION

Submitted in Partial Fulfilment
of the Requirements for the
Degree of

MASTER OF SCIENCE
(Electrical Engineering)

at the

UNIVERSITY OF ULSAN

by

Md. Niaz Morshedul Haque

October 2021

TSCH-Based Scheduling for Throughput and Delay Optimization of IEEE 802.15.4e: A Deep Learning-Based Approach

Approved by Supervisory Committee:

[illegible]

Prof. Hyung-Yun Kong, Chair

[illegible]

Prof. In-Soo Koo, Supervisor

[illegible]

Prof. Sangjo Choi

School of Electrical, Electronic and Computer Engineering
University of Ulsan, South Korea
Date: October, 2021

VITA

Md. Niaz Morshedul Haque was born in Bangladesh. He received the B.Sc. degree in electrical and electronic engineering (EEE) from the Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh, in 2012.

Since 2019, he is pursuing the Master degree from the University of Ulsan, South Korea, under the supervision of Professor In-Soo Koo. He served as a network operation center (NOC) engineer in the telecom industry from 2012 to 2013. He joined the Leading University (LU), Sylhet, Bangladesh, as a Lecturer in 2015. His current research area includes the industrial internet of things (IIoT), resource allocation of IEEE 802.15.4e TSCH networks, and deep learning-based scheduling scheme.

Dedicated to Almighty Allah (for His countless blessings)

and

My family

(for their prayers, love, and support)

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Prof. In-Soo Koo of the School of Electrical, Electronic and Computer Engineering at the University of Ulsan for his supervision, advice, continuous support, and constant encouragement throughout my Master study and related research. His patience, motivation, immense knowledge, and continuous guidance helped to conduct my research and writing of this thesis. Besides my advisors, I am gratefully indebted to my other thesis committee members: Prof. Hyung-Yun Kong, Prof. Sangjo Choi, for their efforts to go through my thesis, insightful comments, and suggestions.

I would like to thank all members of the multimedia communication system laboratory (MCSL) for their friendship and support. I am very much grateful to the University of Ulsan for giving me an excellent research environment and financial support. My special thanks go to BK21 Plus Program for the financial support throughout my Master's studies.

I would like to express my heartiest thanks to all members of Bangladeshi community in Ulsan, South Korea for their love, respect, and support.

Finally, I am ever grateful to my parents Late Motaharul Haque and Naznin Haque for raising me and helping me to achieve all that I have in my life. I am also grateful to my wife Tamanna Hossain for her support and all the sacrifices, she has made, for my achievement.

MD. NIAZ MORSHEDUL HAQUE
October 2021
University of Ulsan

ABSTRACT

TSCH-Based Scheduling for Throughput and Delay Optimization of IEEE 802.15.4e: A Deep Learning-Based Approach

By

Md. Niaz Morshedul Haque

Supervisor: Professor Insoo Koo

IEEE 802.15.4e time-slotted channel hopping (TSCH) sets a new standard for the industrial internet of things (IIoT) due to its simple architecture and productiveness for enhancing credibility in ultra-low-power absorption of industrial appliances. The performance of TSCH is also mainly dominated by the media access control (MAC) mechanism, which consists of refitment, enumeration, composition, and patronization of data transmission schedules that are not accurately prescribed. Most researchers are trying to establish many pragmatic scenarios. Their main approach is to schedule TSCH networks in a centralized way while framing scheduling problems as the nature of throughput and delay in the network.

The main approach of this dissertation is to find a quicker and more exact solution for the scheduling of the TSCH network. We utilize the benefits of a deep learning scheme to reduce the execution time of IEEE 802.15.4e TSCH network scheduling.

Firstly, we propose a Hungarian-based scheduling solution for TSCH networks

by considering throughput and delay with fairness. The scheme proposed previously considered the only throughput for a TSCH network. We utilize maximum link weight alignment in a bipartite graph for TSCH networks to constitute the frames' cell scheduling. In this dissertation, the weight of the bipartite graph is computed by considering both network throughput and delay. Here, we incorporate a window concept to determine moving average network throughput and delay. The throughput and delay parameters are multiplied by the corresponding moving average throughput and delay values to ensure fairness in the bipartite edge weight.

Secondly, we propose a deep learning-based DNN scheme to reduce the execution time of scheduling. The proposed DNN scheme uses the Hungarian solution's training data. When the proposed DNN scheme accepts the weight of the bipartite edge as input, it will offer cell assignments. The proposed DNN scheme is remarkably accurate by learning the relationships between the Hungarian scheduling algorithm's input and output. As a result, it provides quick and precise rational results compared to the Hungarian-based scheduling algorithm.

Thirdly, we design a scheduling method considering a TSCH network in coexistence method of interference network cluster (INC). The proposed dual-stage Hungarian-based scheduling method can do the transmission schedule of the TSCH network by avoiding collision from INC and made the throughput maximization of own network with minimizing the INC throughput. The learning-based DNN scheme is also utilized for reducing the execution time of scheduling.

Contents

List of Figures

List of Tables

Nomenclature

ASN	Absolute slot number
CSI	Channel state information
DL	Deep learning
DNN	Deep neural network
HG	Hungarian
INC	Interference network cluster
IoT	Internet of things
IIoT	Industrial internet of things
ISM	Industrial, scientific, and medical
LCG	Linear congruential generator
MAC	Media access control
MSE	Mean squared error
PDF	Probability density function
ReLu	Rectified linear unit
TSCH	Time slotted channel hopping

Chapter 1

Introduction

1.1 Motivation

The development of technologies has accelerated the expansion of the industrial internet of things (IIoT). It provides an unprecedented potential for participants in various industries. IEEE 802.15.4e is a rectification to the medium access control (MAC) protocol established by IEEE 802.15.4, the pioneer standard of the internet of things (IoT). TSCH is a feature of the IEEE 802.15.4 standard that allows appliances to accommodate ample industrial applications.

The IEEE 802.15.4e TSCH overcomes the drawbacks of IEEE 802.15.4 by delivering excellent credibility and low consumption of power to various applications of industries in extreme conditions. Furthermore, it improves network durability by adding significant redundancy and increasing communication path by reducing interference and multipath fading effects. Moreover, it is possible by utilizing the capabilities of channel hopping and time allocation. TSCH's channel speed is the main component to its higher reliability, and it has been called the "heart" of industrial low power wireless schemes like Wireless HART ISA100.11a, and IETF 6TiSCH.

The TSCH is a deterministic network, meaning that the actions that occur are well-known in each time slot, and a schedule controls communication. Despite its significance, the standard only specifies the procedures for executing a communication schedule, how the schedule is created, progressed, or managed. The scheduling method, or allocating links to cells for data transmission, is a fundamental feature of IEEE 802.15.4e. It can be centralized or distributed, but it develops gingerly and according to its specific application requirement. A network link follows a scheduling method that instructs what is happening on each slot: send, receive, or idle. The question of how to construct a schedule is still a burning issue and find a solution.

In the pragmatic scenario, scheduling in TSCH specifies the frequency and slots for each link of a node. In recent years, many researchers have addressed the scheduling problem for the TSCH protocol, from centralized to distributed solutions.

1.2 Thesis objective

The main objective of this research is to incorporate a deep learning scheme to TSCH network scheduling and utilize the advantages of DL to reduce the execution time of scheduling. TSCH network schedules for links to cell assignment of a slot-frame can be constructed as a maximum weighted bipartite matching approach. In this research, we design bipartite weight composed of throughput and delay parameters, and we use the Hungarian algorithm for proper cell assignment. We also design a dual-stage Hungarian-based scheduling scheme that can smartly avoid the collision from INC. For both cases, the training data is generated with the Hungarian scheduling algorithm and train a deep neural network (DNN) accordingly. The simulation results show that the proposed deep learning-based scheduling scheme can provide performance similar to Hungarian algorithm–based schemes but with low execution time.

1.3 Thesis outline

This thesis consists of four chapters as follows:

- **Chapter 1** presents motivation, thesis objective, and thesis outline.
- **Chapter 2** provides a technique, deep learning-based scheduling scheme for IEEE 802.15.4e TSCH networks
- **Chapter 3** propose a method of TSCH-based scheduling method of IEEE 802.15.4e in coexistence with INC: DNN Approach
- **Chapter 4** concludes the thesis contributions and future works.

Chapter 2

Deep Learning-Based Scheduling Scheme of IEEE 802.15.4e TSCH Networks

2.1 Introduction

The internet of things (IoT) is gradually increasing in popularity due to its multi-functionality 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] as an extension of IEEE 802.15.4 [4], which can use media access control (MAC) functionality to invoke the demands of industrial applications and operations [5]. Time-slotted channel hopping (TSCH) is a basic MAC protocol for IEEE 802.15.4e. TSCH combines multi-channel time slot schedule access (in units of slot-frames or super-frames) 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 be centralized or not; however, it needs to be established based on the applications' tangible demands. 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 in network connections. Diverse channels are attributed 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 scheduling specifies the frequency and slots for every link of a node. Currently, we can see that most research personnel have fixed their scheduling under the TSCH protocol [11]. Scheduling algorithms depend upon non-causal information about instantaneous channel qualities, such as the signal-to-noise ratio (SNR) [12,13]. Some other related 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) [14-16]. In a real scenario for wireless communication networks, channel state information (CSI) is impacted by several factors in a deterministic and random process, 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 chapter shows the scheduling problem as maximization of edge weights based on throughput and delay effects. Here, both parameters consider radical assumptions regarding CSI. We consider the main problem to be the scheduling of cells for links in the TSCH network. A TSCH-based scheduling algorithm is executed by computing a bipartite graph as the vertex of the upper side, with all subgroups of non-interfering links and slot-frame matrix cells as the lower vertex. The edge weights consider the summing of normalized throughput and delay to ensure a maximized edge weight with fairness for a bipartite graph (details are discussed in Section 3) [19]. The throughput provides the maximum data transfer, and delay is minimized to ensure the reliability of network. The Hungarian algorithm performs cell assignment by adequately understanding the 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 the state-of-the-art strategy for numerous issues at present [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, including electric load forecasting [22], prediction of energy elements [23,24], theft detection of electricity [25], energy storage system [26] and distinct field of wireless communications [27-29].

Numerous recent reforms have concentrated on intelligent interactions to reap significant potential benefits. DL is also used to achieve beneficent performance over traditional methods in diverse current work in the wireless communication context [30-32]. Several numerical optimization solutions to solve signal processing tasks have already been suggested by distinguish scholars [33-35]. 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 of deep learning and to improve data efficiency by using deep learning [36-40]. We got the motivation from research in [41], in which the authors provided an outline for applying deep learning technology to the allocation of wireless resources. Here, we address the constraints of conventional optimization approaches and the scope of DL paradigms in wireless networks. Deep learning also plays a tremendous role in increasing the quality of service (QoS) on the internet of things [42]. Deep learning in radio communications for cloud computing, such as the design of a training system for end-to-end wireless communications, has demonstrated that it can outperform traditional wireless communications. The low productivity from training time in 5G wireless networks and communication systems is a constraint when implementing wireless system neural networks [43]. Deep learning is not yet sophisticated in wireless communications. 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, managed services, and quality enhancement [41,43]. Deep learning can assist communication networks with complex operating conditions by accelerating massive amounts 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 [44]. Authors have 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 [45].

The proposed DNN is trained to learn non-linear mapping between optimal network parameters, such as time slot allocation and transmit power [39]. Training data were created using an iterative algorithm that is based on sequential parametric convex approximation (SPCA). A DL-based scheme to estimate channel parameters for wireless resource transfer was described in [46]. Based on input from the energy receivers, the channel parameters are set autonomously at the energy transmitter. Authors have exploited the deep learning ability to optimize the effective throughput of wireless networks [47]. They have shown that their extensive experiments are 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 attained high spectral efficiency [48] in less computational time when managing interference.

Based on the discussion above, we intend to incorporate the DL method with IEEE 802.15.4e TSCH network and utilize the benefits of DL methods to reduce the execution time of scheduling. The proposed deep neural network (DNN)-based scheme predicts IEEE 802.15.4e TSCH network scheduling, where valuable information and data are obtained from the Hungarian algorithm–based scheme.

Contributions of the chapter can be summarized as following.

- Firstly, we propose a Hungarian-based scheduling solution for TSCH networks by considering throughput and delay with fairness. The scheme proposed previously is [10] considered the only throughput for a TSCH network. We utilize maximum link weight alignment in a bipartite graph for TSCH networks to constitute the frames' cell scheduling. In this chapter, the weight of the bipartite graph is computed by considering both network throughput and delay. Here, we incorporate a window concept to determine the moving average network throughput and delay. The throughput and delay parameters are multiplied by the corresponding moving average throughput and delay values to ensure fairness in the bipartite edge weight.

- Secondly, we propose a deep learning–based DNN scheme to reduce the execution time of scheduling. The proposed DNN scheme uses the Hungarian solution's training data. When the proposed DNN scheme accepts the weight of the bipartite edge as input, it will offer cell assignments. The proposed DNN scheme is remarkably accurate by learning the relationships between the Hungarian scheduling algorithm's input and output. As a result, it provides quick and precise rational results compared to the Hungarian-based scheduling algorithm.
- Lastly, the efficiency and performance of the DNN scheme are evaluated by simulations. The simulation results show that the proposed DNN scheme can provide similar outcomes to the conventional Hungarian algorithm but with low execution time.

The remainder of the chapter is arranged as follows. In Section 2.2, we demonstrate the system model and different parameters of our proposed model. In Section 2.3, we describe the proposed Hungarian algorithm-based Scheduling scheme and objective function. In Section 2.4, we delineate the learning-based scheme, the architecture of the DL scheme, and the propagation of training and testing data. In Section 2.5, the proposed method's outcomes are illustrated and verified by using simulation. Finally, Section 2.6 concludes this chapter.

2.2 System model

We present a process model that consists of TSCH scheduling, a TSCH network model, a traffic model, a channel model, and a collision graph. Here, we discuss the theoretical concepts, and introduce a mathematical explanation of the contents mentioned above.

2.2.1 TSCH scheduling

Scheduling in time-slotted channel hopping networks 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 and set by the scheduling method. Scheduling details significantly influence network performance, including throughput, node resource utilization, stream latencies, and durability. The cell is considered the key component of scheduling capacity. 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 matching acknowledgment after successful reception.

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

$$f = \mathcal{F}[(ASN + Ch_{offset})\ \% q_h] \tag{2.1}$$

where Ch_{offset} represents the channel offset, q_h is the number of available channels, and $\mathcal{F}$ denotes the mapping function.

2.2.2 TSCH network model

A TSCH network that consists of a single gateway access point is considered, and multiple nodes are represented in a network graph. The nodes are connected by a gateway in order to synchronize each node on the network. In a centralized manner, scheduling is carried out where the scheduler can be based on the network graph, and the network path is determined by applying computational requirements. The scheduler is installed in the gateway, and specifies the allocation of time slots and the frequency for transmissions by each node.

The TSCH network can be designed as $\mathcal{G} = (V, M, d)$ where V is the set of nodes, $V = \{v_1, v_2, \ldots, v_{\mathcal{N}}\}$ in which $\mathcal{N}$ is the total number of nodes, M is the set of links, and d designates 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 $d_{\mathrm{A,B}}$. Each node V_i is configured by a radio while providing a contact range, $\mathcal{R}_i$, greater than the interruption range, $\acute{\mathcal{R}}_i$. The signs $t \in \{1, \ldots, \mathcal{T}\}$ and $f \in \{1, \ldots, F\}$ indicate the time of each slot and the range of network frequencies, respectively; Δ is the length of each slot.

An example of a slot-channel matrix of a simple TSCH network for five-node structure graph $\mathcal{G}$ is illustrated in Figure 2.1. Here, cells are accessed between two non-interference links. In a slot-frame, links such as $C \longrightarrow E$ and $B \longrightarrow A$ both share the same cells; here, the considered position is $(Slot\ offset\ , Channel\ offset) = (4,4)$. This is possible due to the non-interfering connection. Every connection is a computation that appears in the slot-frame cell.

The nodes are presumed to have only one half-duplex radio module, and they use the radio communication system for transmitting or receiving on diverse channels at separate times, but are limited to sending or receiving on a single channel. An independent connection for each slot must be chosen by a scheduling algorithm so that neighboring nodes will not be triggered simultaneously.

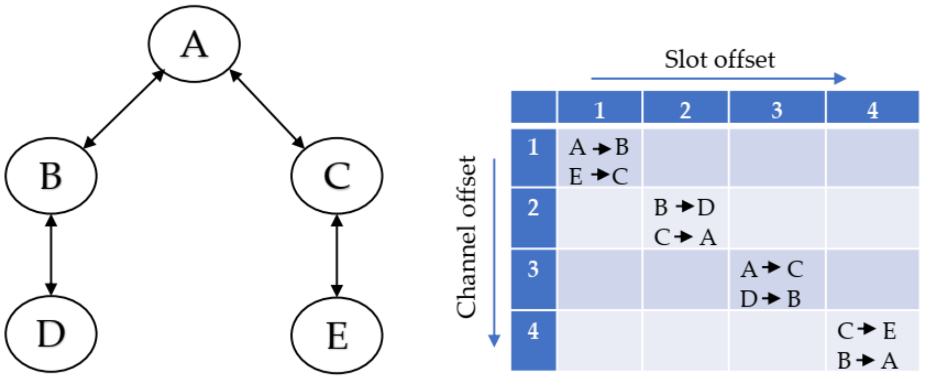


Figure 2.1: A simple network topology for a time-slotted channel hopping slot–channel matrix.

2.2.3 Traffic model

In the sense that each node is still overburdened with traffic, we presume that all nodes are based on a saturated model. A saturated path has flexible traffic so that it obtains a transfer rate that is as fast as it can get for strict timeline requirements without other considerations. To function effectively with this form of traffic, we require optimization of throughput and delay that requires only information on the channel, a thing that we address thoroughly in this work. We are conscious that a node may or may not have data to send when channel is empty under more pragmatic, complex traffic situations, based on the traffic load for each node. As time goes on, we will focus further on the technological challenges associated with this hypothesis, and on preserving the model-free existence of a TSCH scheduler.

2.2.4 Channel model

A combination of channel frequencies $f \in \{1, \ldots, F\}$ and time of the slot $t \in \{1, \ldots, \mathcal{T}\}$ constitutes a cell, $c \in \mathbb{C}(= \{1, \ldots, q_{\mathbb{C}}\})$, where $q_{\mathbb{C}}$ indicates the number of cells. The state of each channel for specific cell c assigned to specific link $m \in M(= \{1, \ldots, q_M\})$ at 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 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 channel gain $H_{m,c}(n)$ will follow Rayleigh fading, and it will be determined by the following probability density function (pdf):

$$F_R(r) = \frac{2r}{\Omega} e^{\frac{-r^2}{\Omega}} \qquad (2.2)$$

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 are calculated, which is considered as the throughput of network. Further it can be computed based on Shannon's formula such that we have.

$$\theta_{m,c}(x_{m,c}(n)) = \frac{\beta}{l} log(1 + \frac{x_{m,c}(n).p}{\beta \eta_o}) \qquad (2.3)$$

where β indicates the bandwidth of the receiver signal, l is the size of the packet, and η_o is the variance of noise.

More specifically, in the proposed scheduling scheme, we consider delay according to the cell position. The relation between delay and delay performance is inversely proportional, which means delay performance is good when delay value is low. If the cell index ($c \in \mathbb{C}$) increases, the delay performance will be decreased. According to our network consideration of the slot frame as shown in Figure 1, the delay will be the same in each slot. Specifically, the delay performance in each cell can be computed with the following equation:

$$\psi_{m,n}^{N}(c) = k_b exp(\left\lfloor\frac{q_{\mathbb{C}}-c}{q_{\mathcal{T}}}\right\rfloor - \left\lfloor\frac{q_{\mathbb{C}}}{q_{\mathcal{T}}}\right\rfloor) \quad (2.4)$$

where N indicates the normalization value of delay performance, $q_{\mathcal{T}}$ is the number of slots, and k_b is a constant parameter for evenly adjusting the unit ratio between throughput and delay.

2.2.5 Collision graph

A collision graph, $\mathbb{Q} = (M, \mathsf{C})$, is defined to consider the interruption in the conceptual model, where C represents the links in collision graph. Its vertex refers to the configuration of graph $\mathcal{G}$. Its edges reveal the conflict between the two links. Figure 2 demonstrates the various ways to transmit data during a collision. The communication mechanism is unicast, but transmission such as $A \longrightarrow B$ and $A \longrightarrow C$ do not occur concurrently, and the collision graph has an edge between them. A legitimate schedule thus allows such data to not be sent through the context of 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 a pair of vertexes in a graph such that there are no edges between either of a couple of vertexes. The purpose of scheduling is that, in a similar communication medium, two colliding nodes are not allocated for transmitting data.

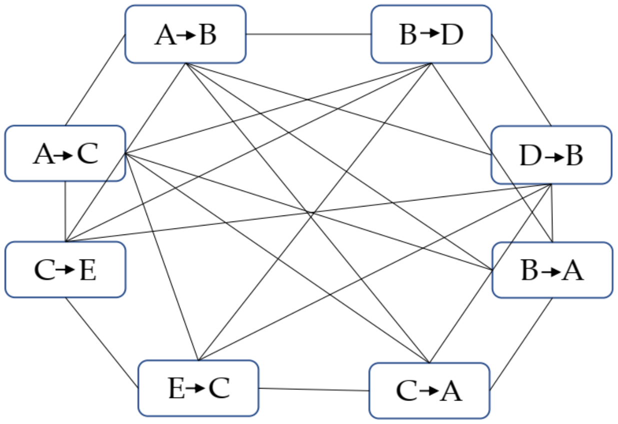


Figure 2.2: Collision graph corresponding to the topology in Figure 1.

2.3 The proposed Hungarian algorithm-based scheduling scheme

In this section, we design and develop the cell scheduling of the frames by utilizing maximum edge weight alignment in a bipartite graph for TSCH networks. Different from the scheme proposed in [10], where only throughput is considered for the weighted bipartite graph, in this paper, bipartite graph weights are computed based on network throughput and delay. After that, we maximize the total network weight as a weighted bipartite graph optimal assignment problem. The vertexes are separated into two dissociated sets (upper and lower) as seen in Figure 2.3. Each subset of the non-interference links obtained from collision graph is a vertex of the top of the bipartite graph and each slot-frame cell is also considered the bottom vertex of the bipartite graph. Thus, the edges connect a vertex from one set to a vertex from another set. A proper assignment is one in which one vertex of the bottom range of the graph absolutely corresponds to any top vertex of the bipartite graph.

Assume $\mathcal{B} = (M, \mathbb{C}, E)$ is a weighed bipartite graph in conjunction with Figure 3. The set of links $M = \{1, \ldots, q_M\}$ is correspond to the collision graph, $\mathbb{Q}$, which is modeled as the upper side of the bipartite graph. The set of cells $\mathbb{C} = \{1, \ldots, q_{\mathbb{C}}\}$ corresponds to the slot-frame matrix cell, $\mathcal{G}$, which is also considered the bottom vertex of the bipartite graph. The set $E = \{e = (m, c) | \, m \in M, c \in \mathbb{C}\}$ consists of the edges of bipartite graph $\boldsymbol{\mathcal{B}}$. The weight of edge $e = (m, c)$ at frame *n* is given by $W_{m,c}(n)$, calculated as follows:

$$W_{m,c}(n) = \alpha u_m^{N0} \theta_{m,c}^N(n) + (1 - \alpha) u_m^{N1} \psi_{m,n}^N \tag{2.5}$$

where α is the weighting factor between throughput and delay. In this paper, we calculate throughput by multiplying normalized throughput $\theta_{m,c}^N(n)$ by the normalized moving average for throughput, u_m^{N0}. Similarly, we calculate the delay by multiplying delay $\psi_{m,c}^N$ by the normalized moving average for delay, u_m^{N1}. We consider this approach to ensure fairness in edge weight.

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

$$\theta_{m,c}^N(n) = \frac{(\theta_{m,c}(x_{m,c}(n)))}{\underset{j \in \mathbb{C}}{\overset{i \in M}{}} max(\theta_{i,j}(x_{i,j}(n)))} \tag{2.6}$$

Moving average throughput u_m^0 and 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 through put and delay use moving-average approaches with consideration for window size k_w. The link-wise moving average throughput, u_m^0, and the delay, u_m^1, are illustrated by the following equations:

$$u_m^0 = u_m^0(n-1) = u_m^0(n-2) + \frac{\theta_{m,c^*}(n-1) - \theta_{m,c^*}(n-k_w-1)}{k_w} \tag{2.7}$$

$$u_m^1 = u_m^1(n-1) = u_m^1(n-2) + \frac{\psi_{m,c^*}(n-1) - \psi_{m,c^*}(n-k_w-1)}{k_w} \tag{2.8}$$

where throughput at frame n is denoted as $\theta_{m,c^*}(n)$ for link m and assigned cell c^*.Similarly, the delay performance at frame n is denoted as $\psi_{m,c^*}(n)$.

Normalized moving averages for throughput u_m^{N0} and delay u_m^{N1} can be obtained with the following equations:

$$u_m^{N0} = 1 - \frac{exp(u_m^0)}{\sum_{i \epsilon M} exp(u_i^0)} \tag{2.9}$$

$$u_m^{N1} = \frac{exp(u_m^1)}{\sum_{i \epsilon M} exp(u_i^1)} \tag{2.10}$$

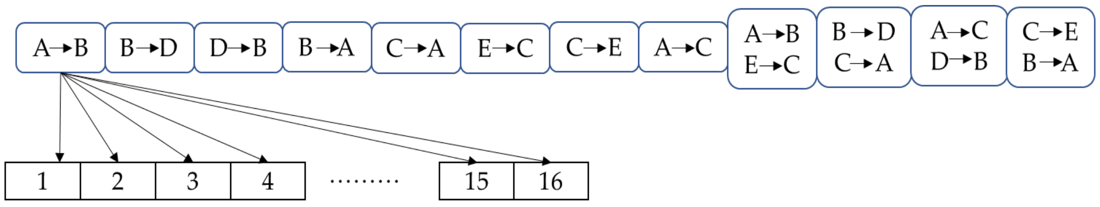


Figure 2.3: Corresponding bipartite graph of our proposed model.

2.3.1 Problem definition and objective function

In a defined channel random distribution, we visualize the problem of weight maximization as a prologue to our strategy. Our objective is to assign corresponding independent links $m \in M$ to cells $c \in \mathbb{C}$ for edges $e \in E$ of all data frames $n \in \mathbb{N}$. This maximizes the total network weight (corresponding to throughput and delay) by sending data packets based on that schedule.

$\overline{\mathcal{W}_T}$ denotes the average weight that is obtained from edges e if a cell is allocated to a link. It is expressed with the following equation:

$$\overline{\mathcal{W}_T} = E\left[W_{m,c}(n)\right] = \frac{1}{\mathbb{N}} \sum_{n\in\mathbb{N}} \sum_{m\in M} \sum_{c\in\mathbb{C}} \xi_{m,c}\, W_{m,c}(n) \tag{2.11}$$

If we characterize ξ as a binary decision variable, it is possible to formulate the mean weight-maximizing process with the following equation:

$$\mathcal{W}_T^* = \lim_{\mathbb{N}\to\infty} {}^{max}_{\xi}\ \overline{\mathcal{W}_T} \tag{2.12}$$

2.3.2 The Hungarian algorithm and solution technique

We use the Hungarian algorithm in graph theory, based on the bipartite graph, to decide the optimal weighted task [23,52]—the Hungarian algorithm which operates in polynomial time. We considered the negligible upper bounds, 2^L(L is the number of links for the topology graph) [9,10]. It performs scheduling tasks based on the maximization of bipartite edge weight. According to our consideration, the whole number of edges of a bipartite graph is the multiplication of the total number of links (top vertexes of the bipartite graph) and the total number of cells (the bottom vertexes bipartite graph). That means the total number of edges of the bipartite graph is. The edge weight is calculated by equation (5). The Hungarian-based scheduling algorithm is shown in Algorithm 2.1

Algorithm 2.1 The Hungarian based scheduling algorithm

```
Step 1: Insert data → Input edge weight from bipartite graph ($(|q_M * q_{\mathbb{C}}|)$)
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: Execute row operation % subtract row minima from each, row-wise
Step 6: Execute column operation % 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, we update the predecessor index
Step 11: Achieve optimal assignment with maximized edge weight
```

Visualization of a single frame's cell assignment is shown in Figure 2.4, and the corresponding edge weight is depicted in Figure 2.5. Figure 2.4 reveals there is no conflict in the assignment. Figure 2.5 illustrates the maximized weight allocation for every corresponding link and cell assignment.

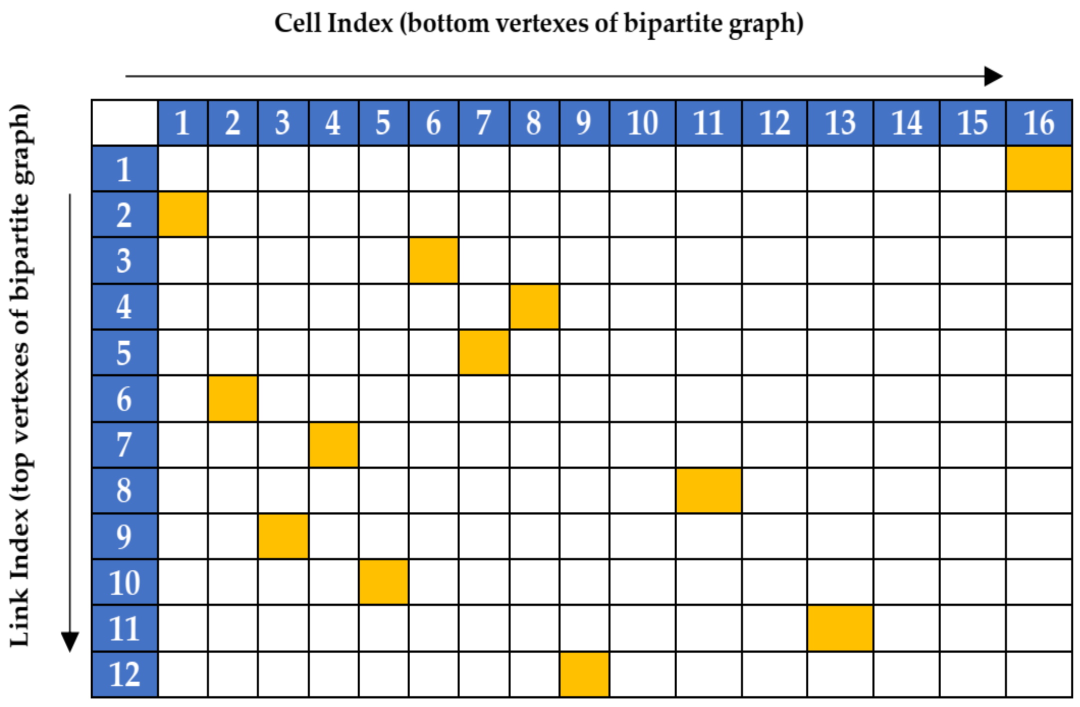

Cell Index (bottom vertexes of bipartite graph)
Link Index (top vertexes of bipartite graph)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12

Figure 2.4: Cell assignment by the proposed Hungarian algorithm–based scheme.

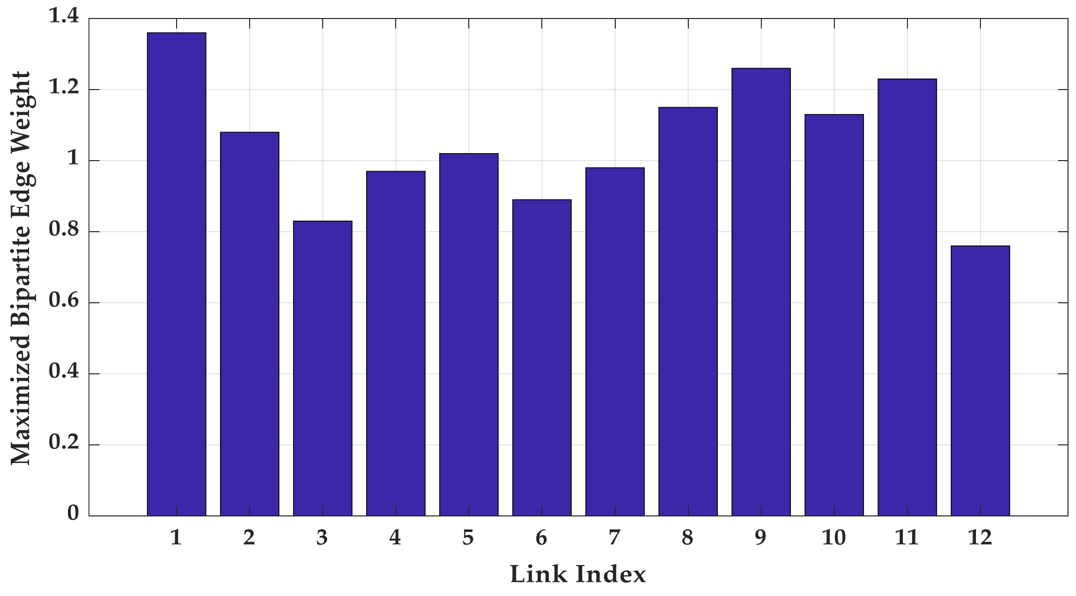


Figure 2.5: Visualization of bipartite edge weight allocation in a single frame.

2.4 Proposed deep learning-based scheduling scheme

In this section, we propose a supervised deep neural network (DNN) that learns the optimal assignment for under-identified CSI. This segment aims to establish a DNN model that can achieve an optimal approximation of the Hungarian algorithm assignment solution for a TSCH network.

2.4.1 Proposed algorithm for generating a data set

Algorithm 2.2 summarizes the proposed Hungarian-based scheduling scheme, and will be used to generate data for DNN training.

Algorithm 2.2 Hungarian-based scheduling to generate DNN training data

```
1: Initialize: constant parameters, $\beta$, $l$, $p$, $\eta_o$, $q_M$, $q_{\mathbb{C}}$, $\alpha$, $k_w$ and $k_b$. Make an enforceable level as the initial point for equation (2.5), for initial moving average throughput $u_m^0(1)$, and for initial moving average delay, $u_m^1(1)$ % needed for first schedule
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 $H_{m,c}(n)$ of the network
5: Update the channel state, $x_{m,c}(n)$
6: Accordingly, determine the throughput, $\theta_{m,c}(x_{m,c}(n))$, with equation (2.3) and delay, $\psi_{m,n}^N(c)$, with equation (2.4) % $\theta_{m,c}(x_{m,c}(n))$ changes for every frame change, but delay is fixed for each frame
```

```
7: Normalized throughput, $\theta^{N}_{m,c}(n)$, is executed with equation (2.6)
8: Normalized moving average throughput, $u^{N0}_{m}$, and delay, $u^{N1}_{m}$, are determined by equation (2.9) and equation (2.10), respectively % the first frame develops by using the initial value of moving average parameters after obtaining the first schedule, and ensures $c^*$; for $n \longrightarrow n+1$, these parameters will be updated with equation (2.7) and equation (2.8), respectively
9: Build an edge weight, $W_{m,c}(n)$, with equation (2.5)
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
```

2.4.2 Structure of the DNN

A simple multilayer perceptron (MLP) is part of the framework of the suggested DNN. MLP is a neural network that is fully connected and comprises one input layer, several hidden layers, and one output layer, as depicted in Figure 2.6. The input of the DNN is bipartite edges $e \in E$, weight $W_{m,c}(n)$ for all links $m \in M$ to cells $c \in \mathbb{C}$, assigned to all data frames $n \in \mathbb{N}$. Here, the set of links is $M = \{1, \dots, q_M\}$, the set of cells is $\mathbb{C} = \{1, \dots, q_{\mathbb{C}}\}$, and the set of data frames is $\mathbb{N} = \{1, \dots, q_{\mathbb{N}}\}$, where q_M, $q_{\mathbb{C}}$, and $q_{\mathbb{N}}$ are, respectively, the number of links, the number of cells, and the number of data frames, or the total number of samples for DNN training. The output of the DNN is cell assignment c^*, which is determined by the Hungarian assignment solution, which means links M

scheduled in cells $\mathbb{C}$. It now ensures that total input to the DNN per data frame is $\{I_1 \ldots, I_n\} = q_M * q_{\mathbb{C}}$. Total output of the DNN per data frame is $\{O_1, \ldots, O_n\} = q_M$.

Neurons in hidden layers and the number of hidden layers is hyperparameters in the DNN and need to be modified to attain maximum accuracy in the optimal solution. The output of each h_{th} layer in the hidden layer can be described by the following equation:

$$\sigma_h = g(\sigma_h w_{h-1} + b_h) \tag{2.13}$$

where w_h and b_h, respectively, are the weights and biases of the h_{th} layer. 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 feature 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 developed for each output of the hidden layer. The activation function explicitly allows the network to acquire non-linear, complicated input and output mapping functions. We use the rectified linear unit (ReLU) activation function in this scheme for the output of any hidden layer. The ReLU is a simple function to implement in DNN, is computationally efficient, and is quicker to calculate [41,42]. 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, $g(r) = r$.

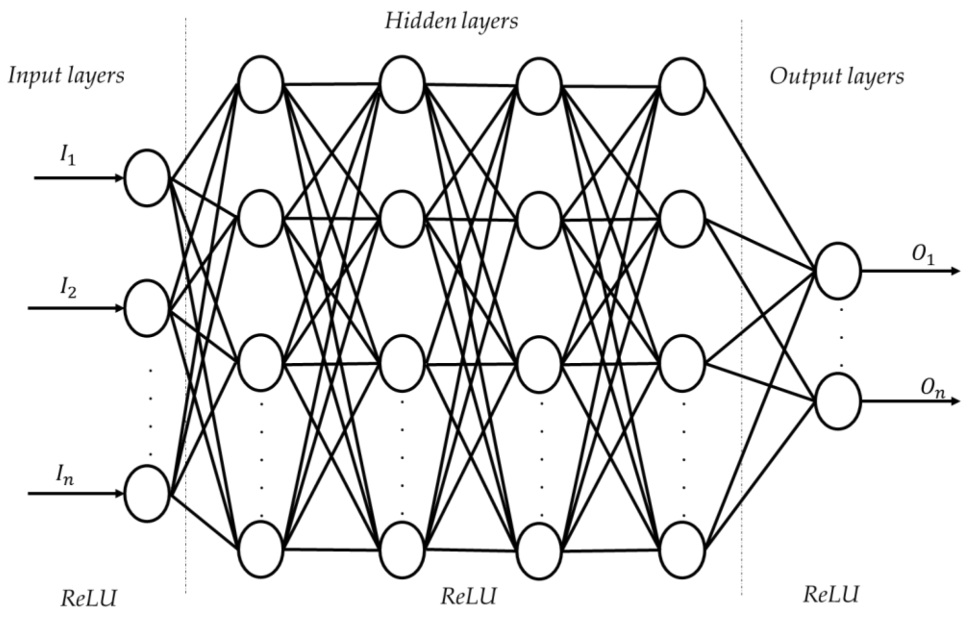


Figure 2.6: The proposed fully connected multi-layer DNN architecture, which comprises one input layer, four hidden layers, and one output layer.

2.4.3 The neural network training mechanism

The essential factor with deep learning is the training of the neural network. A perfectly trained neural network learns the correlation between input and output. In supervised learning, we require enough input and output data frames to train the neural network, from which the DNN recognizes the kinship between input and output. We equip the neural network from training data consisting of samples contained in an iterative method developed using the Hungarian scheduling method as previously discussed. The formation of neural networks is a mechanism by which the loss function is reduced to classify network parameter w. Practically speaking, the loss function is between the actual scheduling, c^*, from the trained data and the DNN output from $\widehat{c^*}$. Thus, the loss function should reduce what we deem to be the mean squared error (MSE) so that the output of the DNN, $\widehat{c^*}$, will approximate the optimal solution, c^*. A well-trained DNN should therefore reduce the following equation:

$$J(w) = \mathcal{E}(|c^* - \widehat{c^*}|)^2 \tag{2.14}$$

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

A total of 10,000 data samples were generated using Algorithm 2.2 based on the Hungarian scheduling method. The training samples were divided into three data segments. To train the DNN, we took 60% of the data, 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 for unbiased assessment that was unknown in the qualified model. This was used to assess how well our scheme learned during training. Testing is an actual, unbiased analysis of the neural network's learned model. To prevent training and validation errors due to the overfitting issue, we employed data normalization. After finishing the DNN training, we could use it for every new value of the bipartite edge weights, $W_{m,c}$, to trace the optimal solution, c^*.

Figure 2.7 shows the training process for the DNN. The network environment generates corresponding data frames or sample data. The offline training is performed under this process and will obtain an optimal interpretation of the IEEE 802.15.4e TSCH network's Hungarian scheduling solutions.

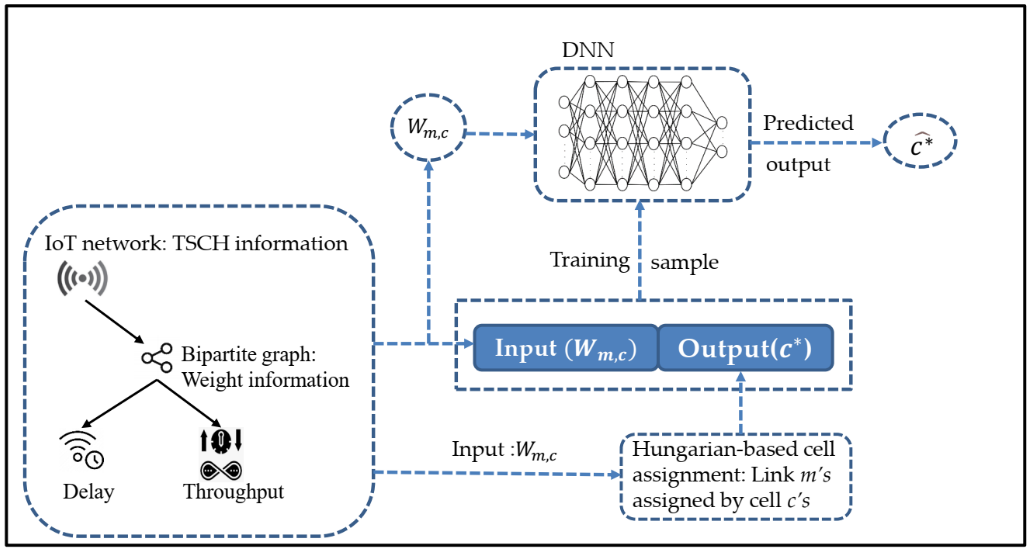

DNN
$W_{m,c}$
Predicted output
$\widehat{c^*}$
Training sample
IoT network: TSCH information
Bipartite graph: Weight information
Delay
Throughput
Input ($W_{m,c}$)
Output(c^*)
Input :$W_{m,c}$
Hungarian-based cell assignment: Link m's assigned by cell c's

Figure 2.7: Training of the DNN.

2.5 Simulation results and performance evaluation

We compare the outcomes provided by the DNN with the iterative Hungarian scheduling method described in the previous section to demonstrate the efficiency of our proposed DNN scheme. For generating data, we used Algorithm 2.2, which projected our whole system model. We generated 10,000 sample data for a different value for α. After obtaining corresponding data samples, we trained the neural network and found the optimal solutions. The TSCH network simulation for data set generation was carried out in MATLAB on a PC with an Intel Core i7 processor and 8 GB RAM. After extracting data samples, the DNN offline training model was executed in Python utilizing the Keras and NumPy libraries.

2.5.1 TSCH network: bipartite network establishment

This work's fundamental goal is to establish a bipartite graph of our proposed model, as described earlier. The model's detailed pedagogy is illustrated in algorithm 2.2 to generate enough sample data to train the neural network correctly. Based on our network model, we considered the parameters in Table 2.1 to establish the described model.

Table 2.1: Parameter specifications for algorithm 2.2

Parameter	Specification
1. Number of links, q_M	*12*
2. Set of links, M	*1:12*
3. Number of cells, $q_{\mathbb{C}}$	*16*
4. Set of cells, $\mathbb{C}$	*1:16*
5. Number of slots, $q_{\mathcal{T}}$	*4*
6. Number of data frames, $q_{\mathbb{N}}$	*10,000*
7. Bandwidth, β	*1 MHz*
8. Bits in each packet l	*1000*
9. Transmission power, p	*10 mw*
10. Noise variance, η_o	*1*

2.5.2 Construction of the DNN model

In this section, we construct a DNN model that requires less computational effort. The developed system can obtain a useful approximation of the Hungarian scheduling method for an IEEE 802.15.4e TSCH network. A DNN with one input layer, multiple hidden layers, and one output layer was considered. The input feature of the neural network is given as bipartite link weight $|W_{m,c}(n)\ |$. 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, $|q_{\mathbb{C}}|$. In our example, the number of input features is $|q_M * q_{\mathbb{C}}| = |12 * 16| = 192$. 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 model are given in Table 2.2. When ReLU activation is used in every hidden layer, a minimal MSE is obtained with little difficulty.

Table 2.2: Parameter specifications for the DNN model

Parameter	Specification
1. Number of input features	*192*
2. Number of hidden layers	*4*
3. Number of neurons in each layer	*800,1600,1200, and 800*
4. Number of outputs	*12*
5. Learning rate	*0.001*
6. Batch size	*100*
7. Number of epochs	*1000*

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

2.5.3 Performance evaluation

2.5.3.1 Determining the model accuracy

Accuracy is one of the most vital parts of the neural network. It indicates how much the proposed model can precisely learn the conventional or previously proposed techniques. We developed an accuracy metric based on the number of assigned cells between traditional or original Hungarian (HG) scheduling and predicted DNN algorithm scheduling. First, we measured accurate parameters. Based on our proposed model, we measure accurate parameters by using the following method. The output of HG scheduling 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.

If $c^* - round(\widehat{c^*}) = 0$, it is accurate, and $c^* - round(\widehat{c^*}) \neq 0$ indicates it is not accurate.

If test samples are $\mathbb{N}_t$, and non-zero or inaccurate parameters are indicated by $\mathbb{P}$, accuracy can be determined with the following equation:

$$Accuracy = \frac{\mathbb{N}_{t*q_M} - \mathbb{P}}{\mathbb{N}_{t*q_M}} * 100\% \qquad (2.15)$$

We measured the accuracy for 1000 test samples as depicted in Table 2.3, such that $\mathbb{N}_t = 1000$. Accuracy can be determined with equation 2.15 for different values of weighting factor α. The significant of α is discussed in the next segment.

Table 2.3: Accuracy of the proposed DNN scheme

Weighting factor	**Accuracy (%)**
α=0.1	*92*
α=0.5	*93*
α=0.9	*92*

2.5.3.2 Bipartite edge weights

Figure 2.8 shows the bipartite edge weights of the previously proposed scheme [10] and the scheme proposed in this paper. As discussed earlier, bipartite edge weight in the earlier scheme was considered the only throughput. One of the contributions of this chapter 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 the bipartite edge weight [10], the maximum value is 1.36, and the minimum value is 0.7. The maximum value of the bipartite edge weight [this paper] is 1.3, and the minimum value is 1.1. As a result, it is shown that our proposed scheme gives more fairness than the scheme in [10].

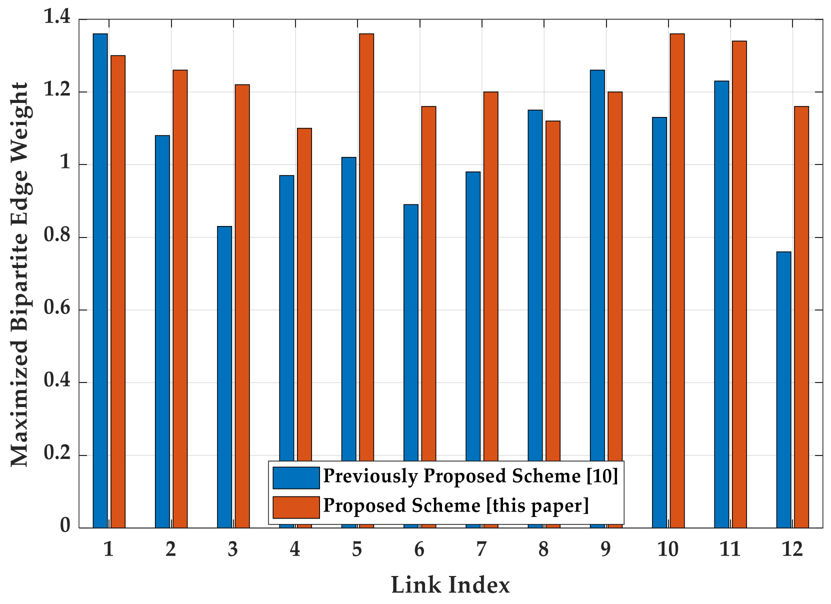


Figure 2.8: The bipartite edge weights from the previously proposed scheme [10] compared to the scheme proposed in this paper.

2.5.3.3 Throughput and delay impact on bipartite edge weight

Figure 2.9 shows the throughput and delay impact on the bipartite edge of 1000 test samples for the Hungarian algorithm (HG) and the DNN. Figure 2.9(a) exhibits the performance of weighting factor α according to the normalized average throughput from HG and the DNN, and Figure 2.9(b) exhibits the performance of weighting factor α according to the normalized average delay from HG and the DNN. We generated several data sets with different values for α and trained the deep neural network. Here, α informs us of the effect on performance from throughput and delay on bipartite edge weights. The significance of weighting factor α is reflected in Table 2.4. It reveals that when weighting factor α is closer to zero, throughput impact is lower on the bipartite edge, and delay performance is higher. If weighting factor α gets close to 1, it indicates throughput impact is more elevated, and subsequently, delay performance is lower. Both parameters exhibit balanced mode for weighting factor α=0.5.

From Figure 2.9 it is also observed that the DNN shows a similar performance and correctly emulate the conventional Hungarian(HG) algorithm. Due to the prediction and accuracy metric (section 2.5.3.1), the DNN is slightly lower than HG.

Table 2.4: The impact of throughput and delay on bipartite edge weight

Weighting Factor	Throughput	Delay Performance
α close to 0	Low	High
α=0.5	Balanced	Balanced
α close to 1	High	Low

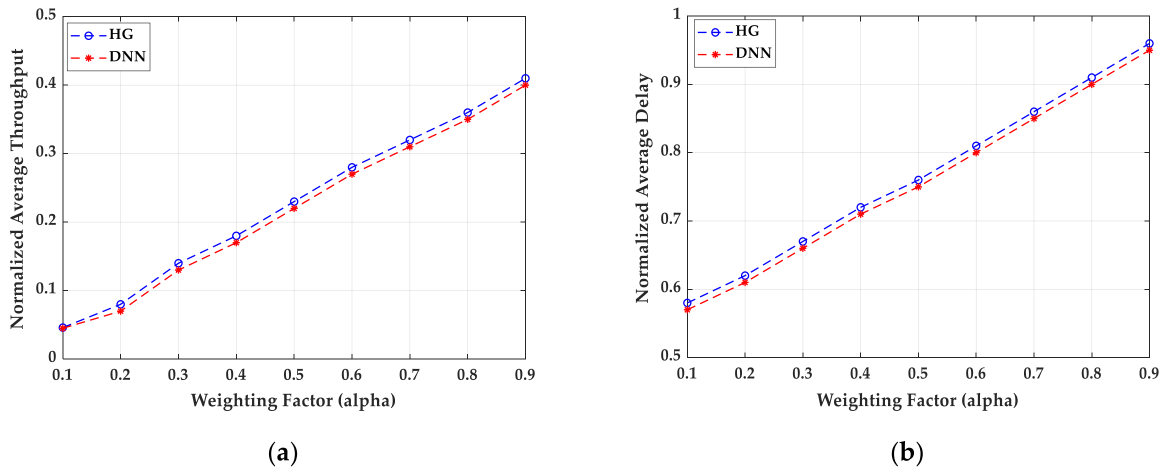


Figure 2.9: The impact of throughput and delay on bipartite links: (**a**) throughput according to weighting factor α; (**b**) delay according to weighting factor α.

2.5.3.4 Cell scheduling

We randomly picked 20 test samples for scheduling and observed that on an average 17 times, our proposed DNN scheme performed similar cell scheduling as the HG scheduling scheme. For the remaining 3 samples, only a few links were differently assigned. Figure 2.10 depicts two examples of the cell scheduling pattern in our proposed scheme. Figure 2.10(a) depicts one of the cases where the scheduling between HG and the DNN is similar for all links. Figure 2.10(b) shows the case where only one link is differently assigned. For HG scheduling, link index 12 is assigned in cell number 12, but for DNN cases, it is assigned in cell number 11, and the rest of the links are assigned accurately for both cases. This scenario supports achieving above 90% accuracy (section 2.5.3.1) for different weighting factor values α.

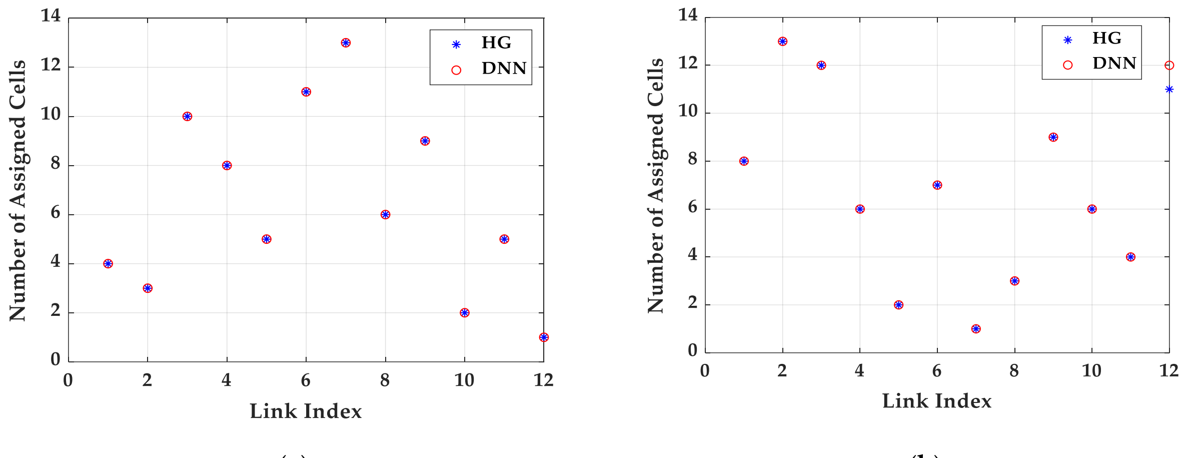


Figure 2.10: Examples of cell scheduling pattern for proposed DNN scheme and HG scheme (**a**) all link indexes in scheduling between the DNN and HG are similar; (**b**) dissimilarity between the DNN and HG for link index 12.

2.5.3.5 Execution time

Finally, the time needed for both schemes HG and DNN, based on the number of data samples, is shown in Figure 2.11. it is observed that the execution time of DNN scheme is much lower than the Hungarian 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.

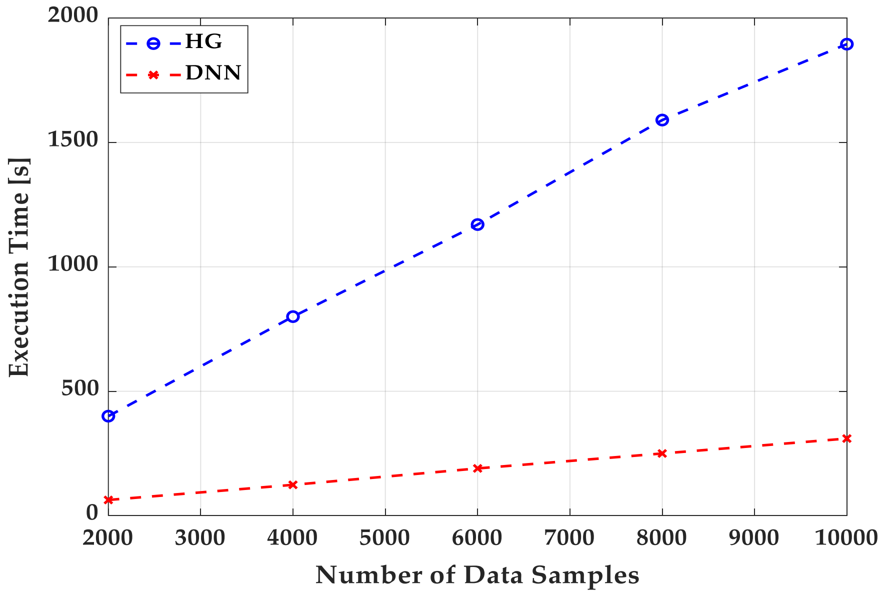


Figure 2.11: Execution time according to the number of data samples for the Hungarian scheme and the DNN scheme.

2.6 Conclusions

In this chapter, we proposed a deep learning–based algorithm for IEEE 802.15.4e TSCH networks for which a DNN was trained to learn the non-linear mapping of bipartite graph parameters, i.e., cell scheduling method, network parameters: throughput, and delay. We executed scheduling based on maximized bipartite edge weight. The data for training were generated by an iterative scheduling algorithm based on the Hungarian scheduling scheme. The simulation results showed that the proposed scheme attained almost the same accuracy as a traditional iterative Hungarian scheduling algorithm. Deep learning–based algorithm has tremendous potential in scheduling in

IEEE 802.15.4e TSCH networks by offering low execution time and more competitive efficiency than a conventional solution. However, if the data size increases, DNN requires more time for training but not for testing. Moreover, the computation time of DNN almost concentrates on training since testing of DNN is composed of simple calculations such as sum and multiplication. Thus, we can easily say that the gap between HG and DNN will increase if the data size increases.

Chapter 3

TSCH-Based Scheduling Method of IEEE 802.15.4e in coexistence with INC: DNN Approach

3.1 Introduction

The scarcity of spectrum is a burning issue over the world. Spectrum scarcity has occurred with the increasing use of the unlicensed industrial, scientific, and medical (ISM) radio bands. According to the expert report, mobile devices have increased from 8 billion in 2016 to 11.5 billion in 2021. As a result, thedata flow from these devices is estimated to grow from 7.2 to 49.0 exabytes per month [50]. In the meantime, the number of internet of things (IoT) application is also growing. The industrial internet of things (IIoT) is a potential application of IoT. IIoT uses the ISM band to develop new technology, establish short-range communication, and implement device-to-device communication [51].

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 slot-offsets. It is gaining acceptance due to its simple architecture and consume low power in industrial application. The performance of TSCH is mainly dominated by the media access control (MAC) mechanism, which covers the refitment, enumeration, composition, and data transmission [52]. The slot frame is the central communication unit of TSCH; a pair of nodes is needed for data transmission. A slot frame is a series of time slots that are repeated continually. A different channel is assigned pseudo-randomly to each timeslot. The schedule specifies which neighbor to interact with and on what channel offset [53].

Due to the excessive use of the unlicensed band, cross-technology interference such as wi-fi conflicts with Bluetooth is another vital issue at present. The IEEE 802.15.4e standard proposes the TSCH mode, which utilizes the channel hopping method to solve these coexistence issues [54,55]. Most of the network technologies of TSCH optimize their own network performance without considering interference from other networks or noise [56]. However, channel hopping from one to the next is not an effective manner since all frequencies face different levels of interference [57,58]. We got some overviews regarding wireless interference patterns with comprehensive analysis about channel selection algorithm with considering random channel model [59].

In this chapter, we present a scheduling algorithm of the TSCH network that can observe the behavior of the surroundings network, especially that make interference with the own network. That is also considered as an interference network cluster (INC) [60,61]. We have proposed a bipartite graph solution of the TSCH network structure. We have proposed dual-stage Hungarian-based scheduling schemes: one for INC and another for our own network and established a relation between them. This will smartly mitigate the collision by maximizing the own network throughput and minimizing INC's throughput.

We have also proposed a learning based DNN scheme that provides us the additional benefit to reduce the computational time of scheduling. We trained the DNN algorithm on a synthetic dataset captured from a Hungarian-based scheduling algorithm. We trained our model offline approach and evaluated the algorithm in a more pragmatic scenario.

The contributions of this chapter are summarized as follows:

- We model a TSCH network by considering INC and formulate algorithms to produce their throughputs and make an interconnection between them.
- We propose a bipartite graph solution for the TSCH slot frame structure. Based on the bipartite graph, we propose a dual stage scheduling algorithm to protect collision between own network and INC, it can maximize the throughput of own network, and minimize the throughput of INC.
- We propose a deep learning-based DNN scheme to reduce the computational time of scheduling tasks. The proposed DNN scheme uses the training data from Hungarian-based scheduling algorithms to avoid a collision from INC. Thus, the proposed DNN scheme correctly emulates the dual-stage Hungarian-based scheduling scheme and reduces the execution time of scheduling.
- The performance of the DNN scheme is verified by Simulation. Furthermore, the simulation results show that the proposed DNN scheme can be similar to a dual-stage Hungarian-based scheduling algorithm for avoiding collision between own network and INC.

To the best of our knowledge, this is the first work that addresses scheduling in IEEE 802.15.4e TSCH networks with considering INC and utilizes the advantages of a learning-based DNN scheme to reduce computational time.

The remainder of the chapter is arranged as follows. In Section 3.2, we demonstrate the system model and problem statement. In Section 3.3, we describe the proposed dual-stage Hungarian algorithm-based scheduling scheme. In Section 3.4, we delineate the training of the DNN scheme and the category of the dataset. In Section 3.5, the proposed method's outcomes are illustrated and verified by using simulation. Finally, Section 3.6 concludes this chapter.

3.2 System model

In this section, we describe the network model, state the mathematical equations for the problem formulation and channel model.

3.2.1 Network model

In this work, we consider that the TSCH network consists of N nodes called the own network and a second unknown network called the interfering network cluster (INC), which may include noise from different sources (e.g., microwaves, noise generators, and jammers) [60,61]. We presume that both networks will have similar priorities and rights to use the spectrum because we focus on unlicensed bands. However, as the INC is unknown, the own network has no idea about what technologies are being used in INC, traffic types, channel state information (CSI), etc. The TSCH network coexistence with INC is illustrated in Figure: 3.1.

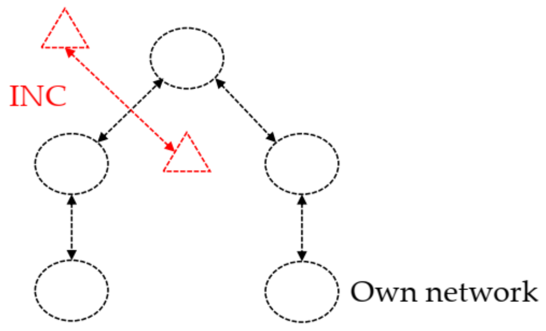
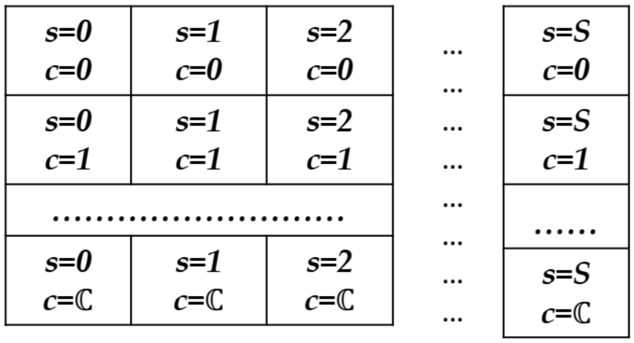


Figure 3.1: Example of TSCH network coexistence with INC.

TSCH uses time and frequency diversity. It's the combination of time division multiple access (TDMA) and frequency-division multiple access (FDMA) techniques. This is core concept of TSCH network. Based on the concept, all N nodes of own TSCH network can communicate with one another on channel $c \in \mathbb{C}$. Each channel is separated into time slots $s \in S$. S time slots and $\mathbb{C}$ channels are available for each slot-frame, $f \in F$ in the TSCH schedule, as shown in Figure 3.2.



s=0 *c=0*	*s=1* *c=0*	*s=2* *c=0*	... ...	*s=S* *c=0*
s=0 *c=1*	*s=1* *c=1*	*s=2* *c=1*	... ...	*s=S* *c=1*
....			... ...	......
s=0 *c=ℂ*	*s=1* *c=ℂ*	*s=2* *c=ℂ*	... ...	*s=S* *c=ℂ*

Figure 3.2: Slot-frame matrix of TSCH network.

3.2.2 Problem formulation

In the model, we consider that all nodes are on their own network because of the number of interfering networks within the INC; any applied protocols features are all unknown. Thus, we only know if one of the nodes of INC has used the channel at a given time. We assume that $P_{s,c}^{f} = 1$ only if the INC used channel *c,* at slot *s*, in frame *f,* at time *(f ,s)*.

We assume that TSCH has a centralized scheduler, and the own network cannot schedule two transmissions, *TX* actions between two nodes at the same times using the same channel. We also consider that INC is strong enough to collide with own network node transmission. We can define that transmission of the own network between two nodes on a channel, *c* at a time (*f, s*), is successful only if own node is transmitting at the same time in that channel and no INC node is transmitting. Every node, $n \in N$ of own network executes a transmission action, $TX_{s,c}^{f,n}$ used channel c, at slot s, for slot-frame f. If own node is transmitting at the time in that channel and no interfering node is transmitting, we use $\gamma_{s,c}^{f,n}$ to determine if an own packet could be successfully delivered or not.

$$\gamma_{s,c}^{f,n} = 1, \; if \;\; TX_{s,c}^{f,n} = 1 \wedge \;\; P_{s,c}^{f} = 0 \tag{3.1}$$

Similarly, the communication of INC is successful if $\gamma_{s,c}^{\prime f,n} = 1$, where:

$$\gamma_{s,c}^{\prime f,n} = 1, \; if \;\; TX_{s,c}^{f,n} = 0 \wedge \;\; P_{s,c}^{f} = 1 \tag{3.2}$$

3.2.3 Channel model

We design and develop a channel model to fulfill the condition of equation 3.1 and equation 3.2. As we discussed earlier, our goal is to develop a bipartite graph-based solution of the TSCH network. For the establishment of the bipartite graph, we need to know the knowledge of throughput. Throughput is the main component for bipartite edge weight, and that help us to solve the scheduling of TSCH [9,10].

As INC is unknown, we don’t know about their channel state information (CSI) and other parameters [60,61]. In this work, we have considered the throughput of INC as a random variable. We use the linear congruential generator for constituting INC throughput. A linear congruential generator (LCG) is an algorithm that generates a series of pseudorandom integers using a piecewise linear equation that is discontinuous. One of the most well-known pseudorandom number generator algorithms is this one [62]. The throughput of INC will be executed by the following equation of linear congruential generator (LCG):

$$Y_i = (aY_{i-1} + q)(mod\ M) \tag{3.3}$$

where, M is modular, $i = cs$, multiplicator, a and q are two suitable chosen integers and the uniform random variable after scaling.

$$U_i = \frac{Y_i}{q} \tag{3.4}$$

Another goal is to minimize the INC throughput for enhancing the reliability of own network. The following equation will minimize the INC throughput:

$$U^* = min \sum_{s \in S} \sum_{c \in \mathbb{C}} U_i \tag{3.5}$$

The own network throughput can be determined by famous Shannon's formula [9,10] depends upon transmission slots for each node:

$$W_i(x) = \frac{\beta}{l} log(1 + \frac{xp}{\beta \eta_o + U^*}) \tag{3.6}$$

where, $i = sc$, channel state $x = X_{c,s} = |H_{c,s}|^2$ $H_{c,s} =$ channel gain, it can be determined as before in chapter 2.

We maximize the own network throughput by following:

$$W^* = max \sum_{s \in S} \sum_{c \in \mathbb{C}} W_i \tag{3.7}$$

3.3 The proposed dual-stage Hungarian based assignment algorithm

In this part, we describe the bipartite graph-based application of the TSCH slot-frame matrix. It proposes a dual-stage Hungarian-based algorithm for transmission scheduling of TSCH network, collision avoiding techniques from INC, maximizing the own network throughput, and minimizing the INC throughput.

3.3.1 Bipartite graph model

According to graph theory-based mathematical structure, a bipartite graph has two independent sets of vertices: top and bottom. Every edge connects a vertex in the top to one on the bottom. It is used for modeling relationships between two different classes of objects [19]. Now, the bipartite graph is a promising application of the TSCH network for presenting an assignment of slots and channels for successful node pair transmission (TX) [9,10]. Throughput is considered as the edge's weight of a bipartite graph [10].

We consider a bipartite graph $\mathcal{B} = (S, \mathbb{C}, E)$ correspond to the slot-frame matrix in Figure: 3.1. The set of channels, $c \in \mathbb{C}$ is the top vertexes of bipartite graph, and the set of slots $s \in S$ is the bottom vertexes of bipartite graph $\mathcal{B}$. The set of edge weights is considered $E = \{e = (s, c) | \, s \in S, c \in \mathbb{C}\}$ as the throughputs of own network and INC (details are in next segment).

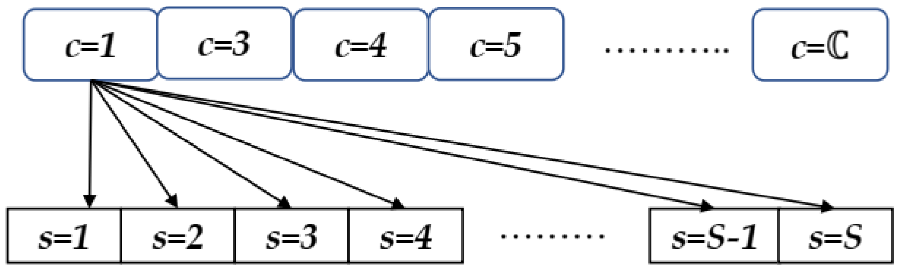


Figure 3.3: Bipartite graph correspond to the slot-frame matrix in Figure: 3.2

3.3.2 The dual-stage Hungarian based algorithm and collision avoidance technique

We propose a dual-stage Hungarian-based assignment technique for avoiding collision between own network and INC. The Hungarian assignment will be performed according to the edge weight of the bipartite graph. We have considered here bipartite edge weights are throughput of own network and INC. In this work, the throughput of INC is computed by linear congruential generator (LCG), and own network throughput is calculated by familiar Shannon's formula (described in section 3.2.3). We considered the negligible upper bounds 2^L (L is the number of node-to-node edge Figure:1)[10].

The number of inputs of the Hungarian algorithm is the total number of edges $S * \mathbb{C}$ of the bipartite graph. Firstly, the INC throughput U_i was considered as the edge weight of the bipartite graph. The Hungarian algorithm performed the assignment according to the minimization of edge weight of the bipartite graph and we obtained the $P_{s,c}^{f}$ and minimization of INC throughput, U^*. Secondly, the own network throughput W_i will be calculated based on the U^*, which was considered as the bipartite edge weight. The Hungarian again performed the scheduling task based on maximization of bipartite edge weight and we obtained the transmission of own network, $TX_{s,c}^{f,n}$ and maximized throughput of own network, W^*. Though most of the collisions between own network and INC were mitigated by this technique, every transmission of own network $TX_{s,c}^{f,n}$ was checked channel by channel and slot by slot. If INC used, $P_{s,c}^{f}$ the specifically assigned slot by own network, that means the duplicate schedule had found in that time own network shifted their scheduling on the following slot based on the second highest edge weight value.

The proposed algorithm is given below:

Algorithm 3.1: Dual-stage Hungarian-based scheduling scheme

```
1: Begin
2: //INITIALIZATION
3: for each $c \in \mathbb{C}$, $s \in S$ and $f \in F$:
4:     Run linear congruential generator (LCG) to obtain INC throughput $U_i$
5:     Run Hungarian algorithm based on minimization $U_i$ of bipartite edge weight

       %where INC throughput used as bipartite edge weight
6:     Obtain $P_{s,c}^{f}$ and $U^*$
7:     Own network throughput, $W_i$ can be calculated
8:     Run Hungarian algorithm based on maximization $W_i$ of bipartite edge weight

       % where own network throughput used as bipartite edge weight
9:     Obtain $TX_{s,c}^{f,n}$ and $W^*$
10:    if $TX_{s,c}^{f,n}$=$P_{s,c}^{f}$,
11:        else $TX_{s,c}^{f,n}$ happen in the assigned slot according to step 8
12:    end if
13:    ensure: $TX_{s,c}^{f,n} = 1 \wedge P_{s,c}^{f} = 0$ and $TX_{s,c}^{f} = 0 \wedge P_{s,c}^{f} = 1$
14: end for
15: //MAIN LOOP:
16: Generate 10,000 data frames for DNN training
```

3.4 The proposed deep learning based DNN scheme

In this section, we propose a deep learning-based supervised DNN scheme that accepts the training data from a dual-stage Hungarian-based scheduling scheme (Algorithm 3.1). The proposed DNN scheme performs the scheduling by learning the kinship between the input and output of the scheduling algorithm of its own network. That means input is the maximized throughput of own network (bipartite edge weight), and the output is the transmission scheduling of own network, based on the learning DNN performs the optimal scheduling of own network.

The structure of the DNN is the same as in chapter 2. In this part, we will discuss the dataset division and training procedure of the proposed DNN scheme.

3.4.1 Dataset distribution and training of DNN

We have needed enough sample data for correctly training the DNN. We generated a total of 10,000 data frames using algorithm 2 based on a dual-stage Hungarian-based scheduling algorithm. We considered three parts of the dataset to train the DNN. We fed 60% of the data for training, kept 20% for validation and 20% for testing. Validation is the unbiased method, that is, assessment procedure as to how well our model learns the knowledge during training. Testing is the actual procedure for analyzing neural network performance. We employed data normalization techniques for avoiding overfitting issues. When the DNN training was finished, we used the scheme for every new value of the own network throughput, W_i to find the optimal solution of, $TX_{s,c}^{f,n}$.

Figure 3.4 describes the training process of DNN. The network environment generated enough samples of data frames based on TSCH information. The offline supervised training was performed based on these data samples. We obtained the optimal interpretation of dual-stage Hungarian-based transmission scheduling, and DNN contributed to reduce the computational time of execution.

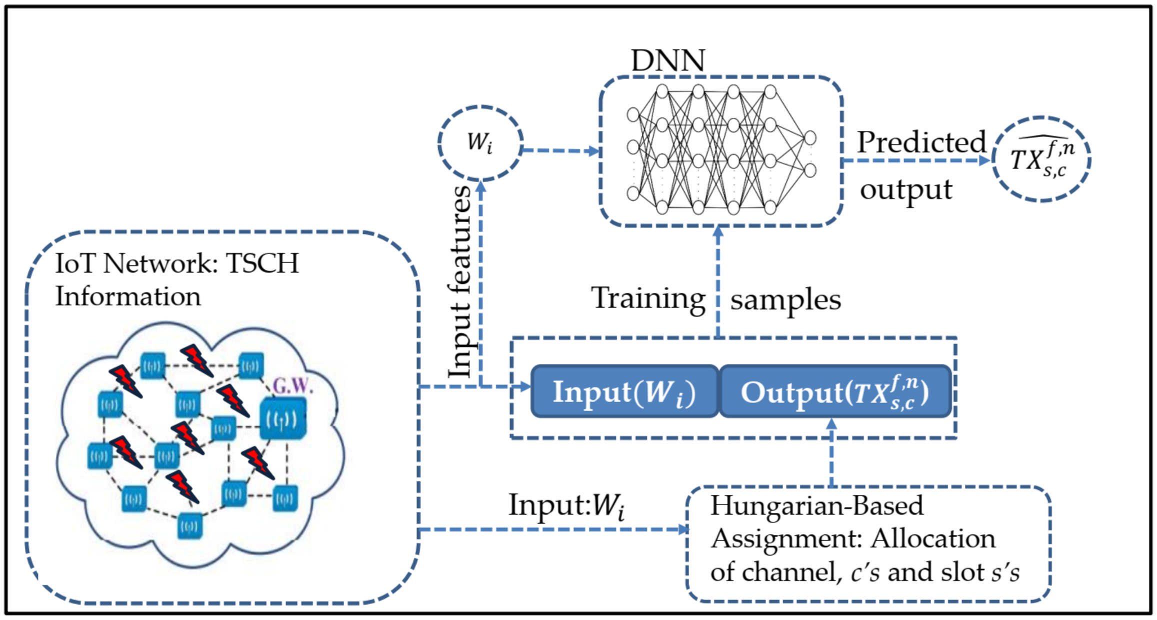

DNN
W_i
Predicted output
$\widehat{TX_{s,c}^{f,n}}$
Input features
IoT Network: TSCH Information
G.W.
Training samples
Input(W_i)
Output($TX_{s,c}^{f,n}$)
Input:W_i
Hungarian-Based Assignment: Allocation of channel, c's and slot s's

Figure 3.4: Training of our proposed DNN scheme.

3.5 Performance evaluation

We compared the outcomes of DNN with the Hungarian-based scheduling algorithm to demonstrate our proposed scheme's efficiency. Here the simulations were performed in two steps: first, we executed algorithm 1 for generating enough samples data; second, after obtaining data samples, the DNN training was performed. Algorithm 3.1 for data set generation was carried out in MATLAB on a PC with an Intel Core i7 processor and 8 GB RAM. After getting data samples, the DNN offline training was executed in Python utilizing the Keras and NumPy libraries.

3.5.1 TSCH network model coexistence with INC

The primary goal of this chapter is to design scheduling of the TSCH network model to avoid Collison from INC. The details of the model are described in Algorithm 3.1. The DNN training data is generated from this model to train the DNN accurately. The following parameters are considered in Table 3.1 for establishing the network model in Algorithm 1.

Table 3.1: Parameter Specifications for Algorithm 1

Parameter	Specification
1. Number of channels, $\mathbb{C}$	*12*
2. Number of slots, S	*16*
3. Number of data frames, F	*10,000*
4. Bandwidth, β	*1 MHz*
5. Bits in each packet l	*1000*
6. Transmission power, p	*10 mw*
7. Noise variance, η_o	*1*

3.5.2 Building a DNN scheme

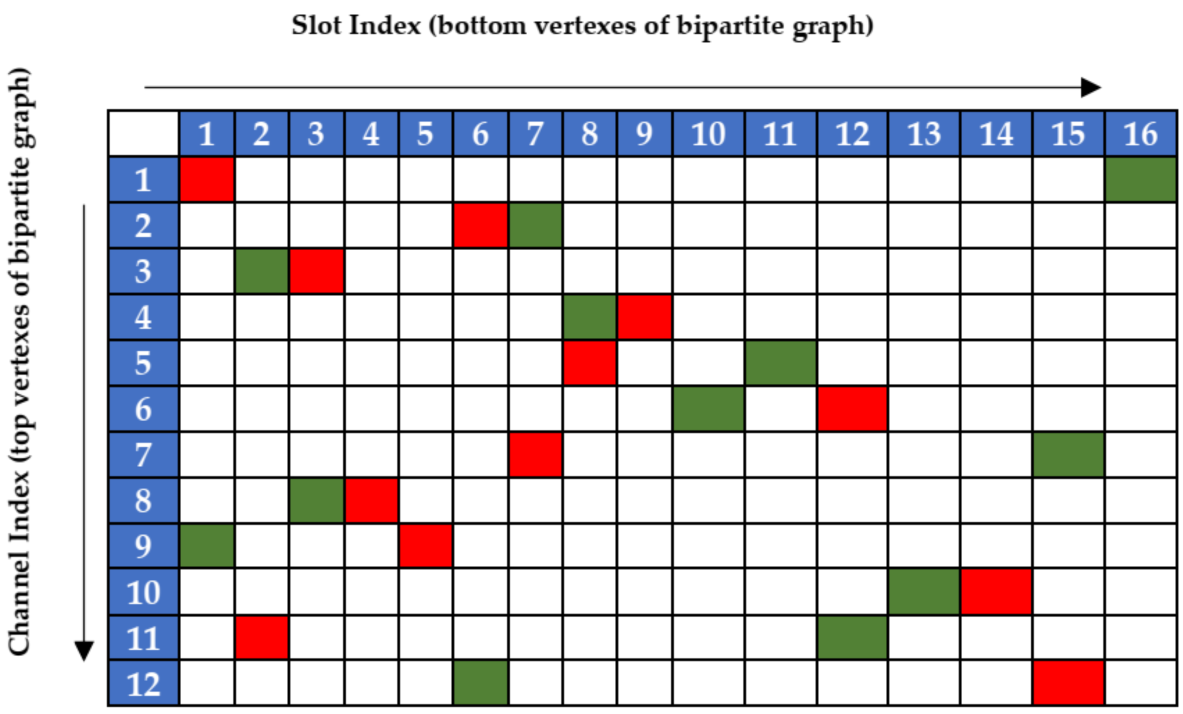
We have considered a DNN a model with one input layer, multiple hidden layers, and one output layer. The number of input features of DNN is the bipartite edge weight, which means the throughput of own network. The total number of bipartite edge weights is the multiplication of the total number of channels and the total number of slots, $\mathbb{C} * S$. The output is considered how many times the own network uses the channels; it means the total number output of DNN is the total number of channels, $\mathbb{C}$. We have used the rectified linear unit (ReLU) activation function for every hidden layer to obtain minimal min square error (MSE) with little difficulty. The specification of the proposed DNN scheme is given in Table 3.2.

Table 3.2. Parameter Specifications for the DNN scheme

Parameter	Specification
1. Number of input features	*192*
2. Number of hidden layers	*4*
3. Number of neurons in each layer	*800,1600,1200, and 800*
4. Number of outputs	*12*
5. Learning rate	*0.001*
6. Batch size	*100*
7. Number of epochs	*1000*

3.5.3 Assignment method

Figure 3.5 illustrates the transmission, $TX_{s,c}^{f,n}$ scheduling assignment of own network that can avoid a collision from INC. Here green color indicates the transmission assignment of own network executed based on the maximization of own network throughput. The red color indicates that INC uses the slots $P_{s,c}^{f}$; it was executed based on the minimization of INC throughput. Lastly, the own network checked channel by channel and slot by slot. For this frame, we observed that the duplicate assignment was executed on channel 3. Both own network and INC were used slot 3 for the channel 3 but using the above checking method when own network found INC in the same slot; then it shifted its transmission schedule to the next slot 2 based on the second highest of own network throughput.


Slot Index (bottom vertexes of bipartite graph)
Channel Index (top vertexes of bipartite graph)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	🟥															🟩
2						🟥	🟩									
3		🟩	🟥													
4								🟩	🟥							
5								🟥			🟩					
6										🟩		🟥				
7							🟥								🟩	
8			🟩	🟥												
9	🟩				🟥											
10													🟩	🟥		
11		🟥										🟩				
12						🟩									🟥	

Figure 3.5: Assignment method.

3.5.4 Throughput optimization

Figure 3.6 shows the average maximized throughput of own network and average minimized throughput of INC for different channels. Our prosed dual-stage Hungarian-based scheduling scheme can maximize the own network throughput based on the minimization of INC throughput. It increases network reliability for data transmission.

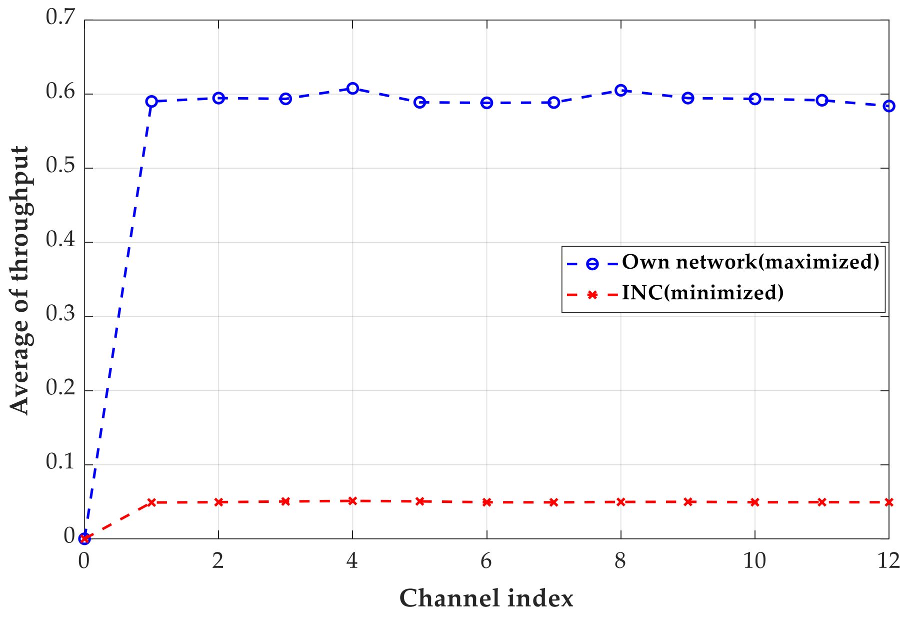


Figure 3.6: Throughput optimization of own network and INC.

3.5.5 Determining the model accuracy

Accuracy determination is the key part of the deep learning-based DNN method. It defines that how much the proposed learning-based scheme learns the original method. We considered an accuracy metric based on the transmission of own network equation 3.8. Firstly, we determined the accurate parameters based on the following method. The transmission assignment of own network $TX_{s,c}^{f}$ is the output of dual-stage Hungarian-based scheme is an integer but predicted output DNN, $\widehat{TX_{s,c}^{f}}$ is not an integer value. For alleviating this problem, we used the round function as follows:

If $TX_{s,c}^{f} - round(\widehat{TX_{s,c}^{f}}) = 0$, it indicates as accurate, and

$TX_{s,c}^{f} - round(\widehat{TX_{s,c}^{f}}) = 0$ indicates it is not accurate.

If test samples are $\mathbb{N}_t$, and non-zero or not accurate parameters are indicated by $\mathbb{P}$, accuracy of DNN can be determined with the following equation:

$$Accuracy = \frac{\mathbb{N}_t * \mathbb{C} - \mathbb{P}}{\mathbb{N}_t * \mathbb{C}} * 100\% \tag{3.8}$$

We measured the accuracy for 1000 test samples as depicted in Figure 3.7. We have achieved around **90%** accuracy of the proposed deep learning-based DNN scheme.

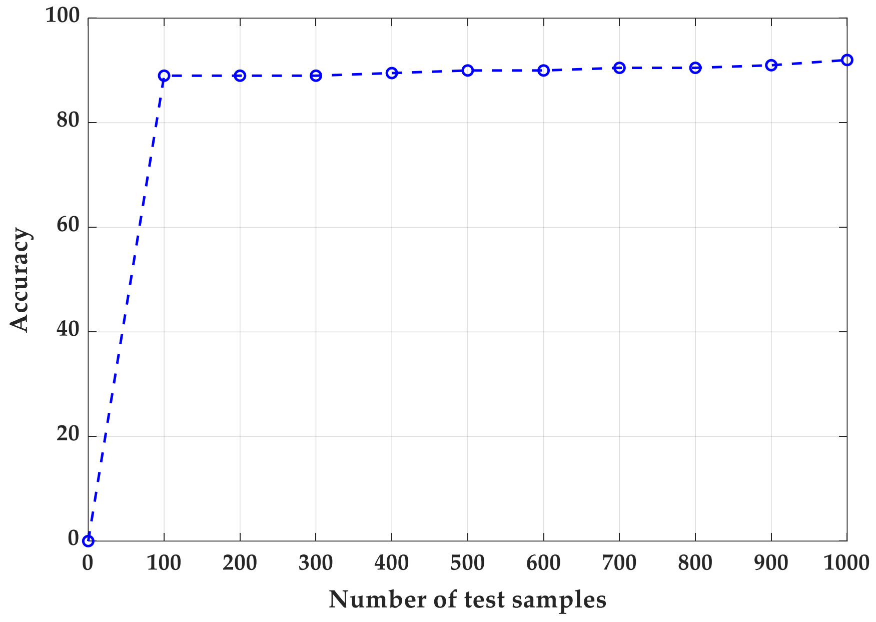


Figure 3.7: Accuracy of deep learning-based DNN scheme

3.5.6 DNN performance

Figure 3.8 and Figure 3.9 show the performance of the proposed DNN scheme based on optimization of throughput for own network and INC. It reveals that the proposed deep learning-based DNN scheme correctly emulates the proposed dual-stage Hungarian-based assignment scheme. DNN is a learning-based scheme and learns the training data from a dual-stage Hungarian (HG) based scheduling scheme; that's why HG shows the optimal and DNN shows the suboptimal for the throughput optimization of both networks.

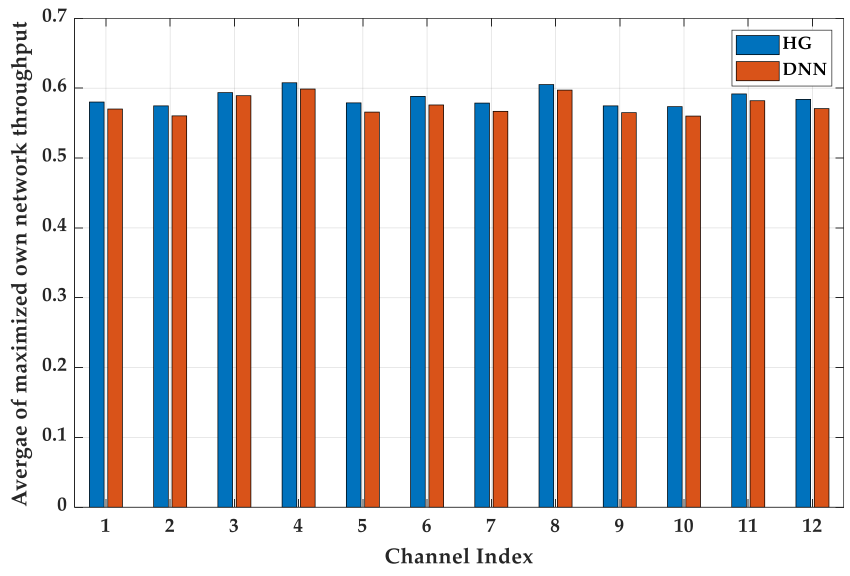


Figure 3.8: Throughput optimization of own network between HG and DNN

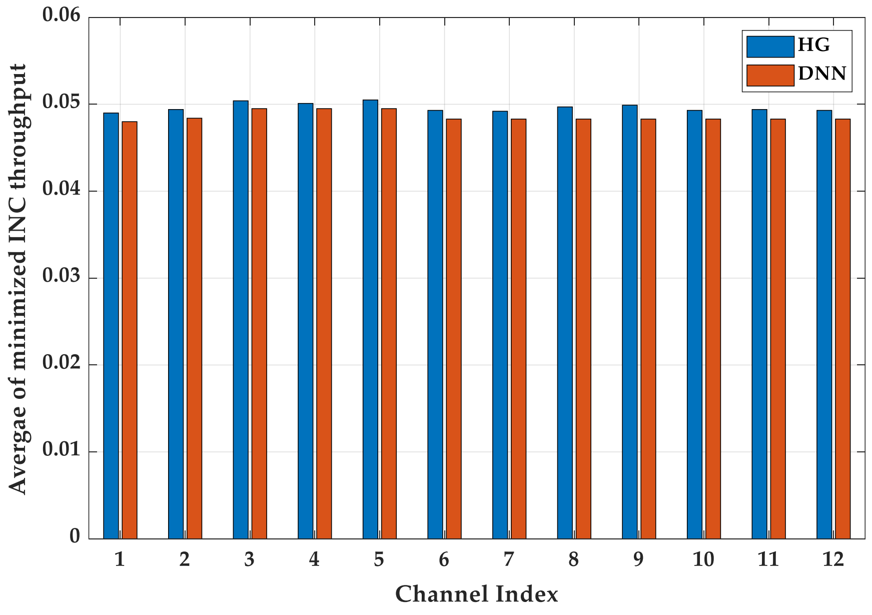


Figure 3.9: Throughput optimization of INC between HG and DNN

3.5.7 Execution time

Figure 3.10 exhibits the execution time for both schemes HG and DNN. It was determined according to the number of data samples. Here observe that the DNN reduced almost 80% execution time than the original method. It is the additional contribution from DNN that can enhance the efficiency of scheduling in the TSCH network.

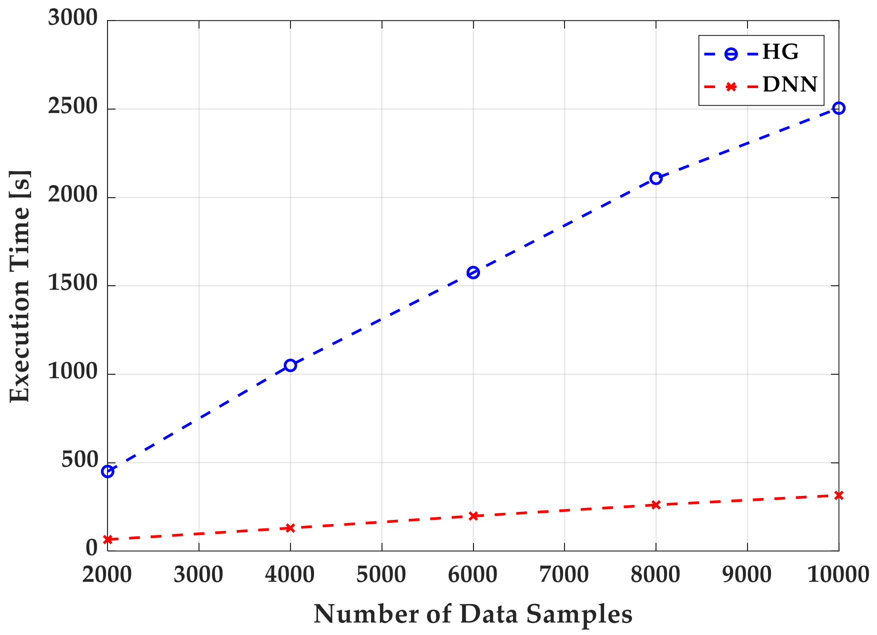


Figure 3.10: Execution time between HG and DNN according to number of data samples

3.6 Conclusion

In this chapter, we proposed a dual-stage Hungarian-based scheduling scheme that can smartly assign the TSCH network without collision from INC. We have also utilized a deep learning-based DNN scheme with this method that provided similar performance to those of assignment techniques and contributed to reducing the execution time of scheduling. The training data set was generated based on a dual-stage Hungarian-based scheduling scheme with maximization of own network throughput and minimization of INC throughput. The DNN was trained by this dataset; it achieved almost 90% accuracy and reduced 80% execution time of original scheme. The deep learning-based DNN scheme showed an efficient performance on TSCH network assignment techniques with less computational time than the original method.

Chapter 4

Summary of Contributions and Future Works

4.1 Introduction

This chapter describes the contributions of this dissertation. The problem statement, objective, methodologies, and results caried out by the proposed solutions are presented in chapter 2 and chapter 3. The first section 4.2 of the current chapter summarizes the primary contributions of those investigations, whereas the outline of the future direction is given in section 4.3.

4.2 Summary of contributions

This dissertation investigated and proposed methods addressing the problem of TSCH based scheduling of IEEE 802.15.4e. It was focused on solving the scheduling problem based on the maximization of throughput with considering delay and moving average throughput and delay to ensure network fairness. It also focused on determining scheduling to avoid collision of interference network cluster (INC) based on the maximization of own network throughput and minimization of INC throughput. The deep learning-based DNN scheme was utilized to reduce the execution time of scheduling.

The contributions of this dissertation, in the context of TSCH based scheduling of IEEE 802.15.4e for a cooperative Hungarian based scheduling algorithm with a deep learning-based DNN scheme, is illustrated below:

- This dissertation considered a TSCH network model for applying the industrial internet of things (IIoT) that can utilize multiple channels and multiple slots for dynamic operations.
- The bipartite graph of the TSCH network proposed that edge weights were composed of network throughput and delay. Here also utilized the moving average throughput and delay for ensuring fairness and moving average delay satisfied the nature of TSCH as the delay between two slot-frames.
- The Hungarian-based algorithm performed the scheduling task based on the maximization of the bipartite edge weight where throughput ensured the maximum data transfer and delay ensured the network's reliability.
- Following that, a dual-stage Hungarian-based scheduling algorithm was proposed. It performed the scheduling smartly by avoiding collision from interference network cluster as well as maximizing the own network throughput based on minimization of INC throughput.
- Furthermore, extensive simulations were conducted to prepare the training dataset and demonstrate the efficiency of the proposed schemes. In both cases, a deep learning-based DNN scheme was utilized that can perform the same as the original scheduling method. It provided an extra contribution to reducing the execution time of scheduling.

- Finally, the performance of the proposed techniques was evaluated by widely used measures such as accuracy, throughput and delay optimization, comparison for execution time, etc. These performance evaluations revealed the efficiency of proposed schemes compared to the original or previously proposed techniques and parameter considerations.

4.3 Future direction

In the future, it is aimed to work on further improvement of the scheduling of the TSCH network. We will identify some other channel parameters that are important for network design. In this work, we consider half-duplex communication of node pair. Furthermore, our target is to consider full-duplex communication. In the future, we will consider uplink/downlink characteristics to enhance network performance and provide optimal scheduling resources and want to apply them in the physical implementation of small applications.

Publications

International Journal

[1] **Under Review: Md. Niaz Morshedul Haque,** Young-Doo Lee, Insoo Koo* "Deep Learning–Based Scheduling Scheme for IEEE 802.15.4e TSCH Networks", Wireless Communications and Mobile Computing.

Journal paper draft under preparation

[2] **To be submitted: Md. Niaz Morshedul Haque,** Young-Doo Lee, Insoo Koo* "TSCH-Bases Scheduling Method of IEEE 802.15.4e in coexistence with INC: DNN Approach"

Bibliography

[1] Ud Din, I.; Guizani, M.; Hassan, S.; Kim, B.S.; Khurram Khan, M.; Atiquzzaman, M.; Ahmed, S.H. The Internet of Things: A Review of Enabled Technologies and Future Challenges. *IEEE Access* **2019**, *7*, 7606–7640, doi:10.1109/ACCESS.2018.2886601

[2] Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Informatics* **2018**, *14*, 4724–4734, doi:10.1109/TII.2018.2852491.

[3] Standard, I.; Society, I.C.; Nguyen, H.Q.; Choi, J.; Kang, M.; Ghassemlooy, Z.; Kim, D.H.; Lim, S.; Kang, T.; Lee, C.G. *IEEE Standard for Local and Metropolitan Area Networks — Part 15 . 4 : Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Computer Society S Ponsored by The*; 2010; Vol. 3600; ISBN 978-1-4244-8858-2.[

[4] IEEE 802.15.4-2006 - IEEE Standard for Information Technology-- Local and Metropolitan Area Networks-- Specific Requirements-- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area N.

[5] Nps, M.; Owner, D.; Medical, N.; Endorsed, S.; Pages, A.; Action, C.; Mccormack, N.C.; Brearley, D.O.N.L. *N Ursing P Ractice S Tandard F Or*; 2010; Vol. 2015; ISBN 9781504408462.

[6] Bae, B.H.; Chung, S.H. Fast Synchronization Scheme Using 2-Way Parallel Rendezvous in IEEE 802.15.4 TSCH. *Sensors (Switzerland)* **2020**, *20*, doi:10.3390/s20051303.

[7] Watteyne, T.; Mehta, A.; Pister, K. Reliability through Frequency Diversity: Why Channel Hopping Makes Sense. *PE-WASUN'09 - Proc. 6th ACM Int. Symp. Perform. Eval. Wirel. Ad-Hoc, Sensor, Ubiquitous Networks* **2009**, 116–123, doi:10.1145/1641876.1641898.

[8] Oh, S.; Hwang, D.Y.; Kim, K.H.; Kim, K. Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH. *Sensors (Switzerland)* **2018**, *18*, 1–25, doi:10.3390/s18041209.

[9] Ojo, M.; Giordano, S. An Efficient Centralized Scheduling Algorithm in IEEE 802.15.4e TSCH Networks. *2016 IEEE Conf. Stand. Commun. Networking, CSCN 2016* **2016**, doi:10.1109/CSCN.2016.7785164.

[10] Taheri Javan, N.; Sabaei, M.; Hakami, V. IEEE 802.15.4.e TSCH-Based Scheduling for Throughput Optimization: A Combinatorial Multi-Armed Bandit Approach. *IEEE Sens. J.* **2020**, *20*, 525–537, doi:10.1109/JSEN.2019.2941012.

[11] Kharb, S.; Singhrova, A. A Survey on Network Formation and Scheduling Algorithms for Time Slotted Channel Hopping in Industrial Networks. *J. Netw. Comput. Appl.* **2019**, *126*, 59–87, doi:10.1016/j.jnca.2018.11.004.

[12] Ojo, M.; Giordano, S.; Portaluri, G.; Adami, D.; Pagano, M. An Energy Efficient Centralized Scheduling Scheme in TSCH Networks. *2017 IEEE Int. Conf. Commun. Work. ICC Work. 2017* **2017**, 570–575, doi:10.1109/ICCW.2017.7962719.

[13] Ojo, M.; Giordano, S.; Portaluri, G.; Adami, D. Throughput Maximization Scheduling Algorithm in TSCH Networks with Deadline Constraints. *2017 IEEE Globecom Work. GC Wkshps 2017 - Proc.* **2018**, *2018-January*, 1–6, doi:10.1109/GLOCOMW.2017.8269114.

[14] Theoleyre, I.H. and F. No TitleAdaptive K-Cast Scheduling for HighReliability and Low-Latency in IEEE802.15.4-TSCH. In Proceedings of the Int. Conf. on Ad Hoc Net. and Wireless (ADHOC-NOW' 18), pp. 3-18; 2018.

[15] Zorbas, D.; Kotsiou, V.; Th, F. LOST : Localized Blacklisting Aware Scheduling. **2018**, 110–115.

[16] Daneels, G.; Latre, S.; Famaey, J. Efficient Recurrent Low-Latency Scheduling in IEEE 802.15.4e TSCH Networks. *2019 IEEE Int. Black Sea Conf. Commun. Networking, BlackSeaCom 2019* **2019**, doi:10.1109/BlackSeaCom.2019.8812869.

[17] Mathew, V.; Manuel, E.M. Cognitive Scheduling in TSCH Based Mobile WSN Using Wavelet Packet Analysis. *2015 IEEE Int. Conf. Signal Process. Informatics, Commun. Energy Syst. SPICES 2015* **2015**, 2–6, doi:10.1109/SPICES.2015.7091440.

[18] Qiu, S.; Chen, D.; Qu, D.; Luo, K.; Jiang, T. Downlink Precoding with Mixed Statistical and Imperfect Instantaneous CSI for Massive MIMO Systems. *IEEE Trans. Veh. Technol.* **2018**, *67*, 3028–3041, doi:10.1109/TVT.2017.2774836.

[19] Ma, J.; Qiao, Y.; Hu, G.; Li, T.; Huang, Y.; Wang, Y.; Zhang, C. Social Account Linking via Weighted Bipartite Graph Matching. *Int. J. Commun. Syst.* **2018**, *31*, 1–14, doi:10.1002/dac.3471.

[20] Date, K.; Nagi, R. GPU-Accelerated Hungarian Algorithms for the Linear Assignment Problem. *Parallel Comput.* **2016**, *57*, 52–72, doi:10.1016/j.parco.2016.05.012.

[21] Samuel, N.; Diskin, T.; Wiesel, A. Learning to Detect. *IEEE Trans. Signal Process.* **2019**, *67*, 2554–2564, doi:10.1109/TSP.2019.2899805.

[22] Bouktif, S.; Fiaz, A.; Ouni, A.; Serhani, M.A. Optimal Deep Learning LSTM Model for Electric Load Forecasting Using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches. Energies 2018, 11, doi:10.3390/en11071636.

[23] Uselis, A.; Lukoševičius, M.; Stasytis, L. Localized Convolutional Neural Networks for Geospatial Wind Forecasting. *Energies* **2020**, *13*, 1–21, doi:10.3390/en13133440.

[24] Praditia, T.; Walser, T.; Oladyshkin, S.; Nowak, W. Improving Thermochemical Energy Storage Dynamics Forecast with Physics-Inspired Neural Network Architecture. *Energies* **2020**, *13*, doi:10.3390/en13153873.

[25] Gong, X.; Tang, B.; Zhu, R.; Liao, W.; Song, L. Data Augmentation for Electricity Theft Detection Using Conditional Variational Auto-Encoder. *Energies* **2020**, *13*, 1–14, doi:10.3390/en13174291.

[26] Desportes, L.; Fijalkow, I.; Andry, P. Deep Reinforcement Learning for Hybrid Energy Storage Systems: Balancing Lead and Hydrogen Storage. *Energies* **2021**, *14*, doi:10.3390/en14154706.

[27] Wang, T.; Wen, C.K.; Wang, H.; Gao, F.; Jiang, T.; Jin, S. Deep Learning for Wireless Physical Layer: Opportunities and Challenges. *China Commun.* **2017**, *14*, 92–111, doi:10.1109/CC.2017.8233654.

[28] Farsad, N.; Goldsmith, A. Neural Network Detection of Data Sequences in Communication Systems. *arXiv* **2018**, *66*, 5663–5678.

[29] Farsad, N.; Goldsmith, A. Detection Algorithms for Communication Systems Using Deep Learning. *arXiv* **2017**.

[30] Lee, W.; Kim, M.; Cho, D.H. Deep Power Control: Transmit Power Control Scheme Based on Convolutional Neural Network. *IEEE Commun. Lett.* **2018**, *22*, 1276–1279, doi:10.1109/LCOMM.2018.2825444.

[31] Wang, S.; Liu, H.; Gomes, P.H.; Krishnamachari, B. Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks. *arXiv* **2018**, *4*, 257–265, doi:10.1109/tccn.2018.2809722.

[32] Zappone, A.; Di Renzo, M.; Debbah, M. Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both? *arXiv* **2019**, *67*, 7331–7376.

[33] Ju, H.; Zhang, R. Throughput Maximization In. *IEEE Trans. Wirel. Commun.* **2014**, *13*, 418–428.

[34] Yang, G.; Ho, C.K.; Zhang, R.; Guan, Y.L. Throughput Optimization for Massive MIMO Systems Powered by Wireless Energy Transfer. *IEEE J. Sel. Areas Commun.* **2015**, *33*, 1640–1650, doi:10.1109/JSAC.2015.2391835.

[35] Li, Q.; Wang, L.; Xu, D. Resource Allocation in Cognitive Wireless Powered Communication Networks under Outage Constraint. *2018 IEEE 4th Int. Conf. Comput. Commun. ICCC 2018* **2018**, 683–687, doi:10.1109/CompComm.2018.8780604.

[36] Bi, S.; Zhang, R.; Ding, Z.; Cui, S. Wireless Communications in the Era of Big Data. *IEEE Commun. Mag.* **2015**, *53*, 190–199, doi:10.1109/MCOM.2015.7295483.

[37] Zappone, A.; Di Renzo, M.; Debbah, M.; Lam, T.T.; Qian, X. Model-Aided Wireless Artificial Intelligence: Embedding Expert Knowledge in Deep Neural Networks for Wireless System Optimization. *IEEE Veh. Technol. Mag.* **2019**, *14*, 60–69, doi:10.1109/MVT.2019.2921627.

[38] Sun, H.; Chen, X.; Shi, Q.; Hong, M.; Fu, X.; Sidiropoulos, N.D. Learning to Optimize: Training Deep Neural Networks for Wireless Resource Management. *arXiv* **2017**, *66*, 5438–5453.

[39] Hameed, I.; Tuan, P.V.; Koo, I. Exploiting a Deep Neural Network for Efficient Transmit Power Minimization in a Wireless Powered Communication Network. *Appl. Sci.* **2020**, *10*, doi:10.3390/app10134622.

[40] Gron, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build1st Ed.; O'Reilly Media, Inc.: Newton, MA, USA, 2017 Intelligent Systems,;*

[41] Liang, L.; Ye, H.; Yu, G.; Li, G.Y. Deep-Learning-Based Wireless Resource Allocation with Application to Vehicular Networks. *Proc. IEEE* **2020**, *108*, 341–356, doi:10.1109/JPROC.2019.2957798.

[42] Kimbugwe, N.; Pei, T.; Kyebambe, M.N. Application of Deep Learning for Quality of Service Enhancement in Internet of Things : A Review. **2021**.

[43] Wu, H.; Li, X.; Deng, Y. Deep Learning-Driven Wireless Communication for Edge-Cloud Computing: Opportunities and Challenges. *J. Cloud Comput.* **2020**, *9*, doi:10.1186/s13677-020-00168-9.

[44] Dai, L.; Jiao, R.; Adachi, F.; Vincent Poor, H.; Hanzo, L. Deep Learning for Wireless Communications: An Emerging Interdisciplinary Paradigm. *arXiv* **2020**, 133–139.

[45] Li, M.; Li, H. Application of Deep Neural Network and Deep Reinforcement Learning in Wireless Communication. *PLoS One* **2020**, *15*, 1–15, doi:10.1371/journal.pone.0235447.

[46] Kang, J.M.; Chun, C.J.; Kim, I.M. Deep-Learning-Based Channel Estimation for Wireless Energy Transfer. *IEEE Commun. Lett.* **2018**, *22*, 2310–2313, doi:10.1109/LCOMM.2018.2871442.

[47] He, D.; Liu, C.; Wang, H.; Quek, T.Q.S. Learning-Based Wireless Powered Secure Transmission. *IEEE Wirel. Commun. Lett.* **2019**, *8*, 600–603, doi:10.1109/LWC.2018.2881976.

[48] Lee, W.; Kim, M.; Cho, D.H. Transmit Power Control Using Deep Neural Network for Underlay Device-to-Device Communication. *IEEE Wirel. Commun. Lett.* **2019**, *8*, 141–144, doi:10.1109/LWC.2018.2864099.

[49] Kuhn, H.W. NThe Hungarian Method for the Assignment Problemo Title. *Nav. Res. Logist.* **2005**, *52*, 7–21.

[50] Index, Cisco Visual Networking, "Global mobile data traffic forecast update, 2016-2021 white paper," Cisco: San Jose, CA, USA, 2017.

[51] Natarajan, R.; Zand, P.; Nabi, M. Analysis of Coexistence between IEEE 802.15.4, BLE and IEEE 802.11 in the 2.4 GHz ISM Band. IECON Proc. (Industrial Electron. Conf. 2016, 6025–6032, doi:10.1109/IECON.2016.7793984.

[52] D. De Guglielmo, S.B. and G.A. Ieee 802.15.4e: A Survey. *Comput. Commun.* **2016**, *88*, 1–24.

[53] Kharb, S.; Singhrova, A. A Survey on Network Formation and Scheduling Algorithms for Time Slotted Channel Hopping in Industrial Networks. *J. Netw. Comput. Appl.* **2019**, *126*, 59–87, doi:10.1016/j.jnca.2018.11.004.

[54] Alkama, L.; Bouallouche-Medjkoune, L. IEEE 802.15.4 Historical Revolution Versions: A Survey. *Computing* **2021**, *103*, 99–131, doi:10.1007/s00607-020-00844-3.

[55] Watteyne, T.; Mehta, A.; Pister, K. Reliability through Frequency Diversity: Why Channel Hopping Makes Sense. *PE-WASUN'09 - Proc. 6th ACM Int. Symp. Perform. Eval. Wirel. Ad-Hoc, Sensor, Ubiquitous Networks* **2009**, 116–123, doi:10.1145/1641876.1641898.

[56] Hammoudi, S.; Harous, S.; Aliouat, Z. External Interference Free Channel Access Strategy Dedicated to TSCH. *IEEE Int. Conf. Electro Inf. Technol.* **2018**, *2018-May*, 350–355, doi:10.1109/EIT.2018.8500259.

[57] Kurunathan, H.; Severino, R.; Koubaa, A.; Tovar, E. IEEE 802.15.4e in a Nutshell: Survey and Performance Evaluation. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 1989–2010, doi:10.1109/COMST.2018.2800898.

[58] D. De Guglielmo, G. Anastasi, and A.S. No TitFrom IEEE 802.15. 4 to IEEE 802.15. 4e: A Step towards the Internet of Thingsle. , *Springer Int. Publ.* **2014**, 135–152.

[59] Ansari, J.; Mähönen, P. Channel Selection in Spectrum Agile and Cognitive MAC Protocols for Wireless Sensor Networks. *MobiWac'10 - Proc. 8th ACM Int. Symp. Mobil. Manag. Wirel. Access, Co-located with MSWiM'10* **2010**, 83–90, doi:10.1145/1868497.1868511.

[60] Mennes, R.; Claeys, M.; De Figueiredo, F.A.P.; Jabandzic, I.; Moerman, I.; Latre, S. Deep Learning-Based Spectrum Prediction Collision Avoidance for Hybrid Wireless Environments. *IEEE Access* **2019**, *7*, 45818–45830, doi:10.1109/ACCESS.2019.2909398.

[61] Mennes, R.; De Figueiredo, F.A.P.; Latré, S. Multi-Agent Deep Learning for Multi-Channel Access in Slotted Wireless Networks. *IEEE Access* **2020**, *8*, 95032–95045, doi:10.1109/ACCESS.2020.2995456.

[62] Tong, Q.; Zou, X.; Tong, H. A RFID Authentication Protocol Based on Infinite Dimension Pseudo Random Number Generator. *Proc. 2009 Int. Jt. Conf. Comput. Sci. Optim. CSO 2009* **2009**, *1*, 292–294, doi:10.1109/CSO.2009.436