TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Shreesottam Kumar Keshari 184611IASM

DECENTRALIZED RESOURCE ALLOCATION IN DEVICE TO DEVICE COMMUNICATIONS

Master`s Thesis

Supervisor: Md Muhidul Islam Khan

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Shreesottam Kumar Keshari 184611IASM

OTSEÜHENDUSES SIDESEADMETE DETSENTRALISEERITUD RESSURSIHALDUS

Magistritöö

Juhendaja: Md Muhidul Islam Khan

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Shreesottam Kumar Keshari

01.04.2020

Abstract

The increasing demand for local services, device to device (D2D) communications are considered an essential component of next-generation cellular networks to improve spectrum reuse, increase bandwidth and system throughput. It enables user equipment to communicate directly with other UE`s. D2D communication in cellular networks is defined as direct communication between two mobile users without traversing the base station. Direct communications between nearby mobile devices will improve spectrum utilization, overall throughput. If the interference produced by D2D users is not properly taken care, there will be deterioration on overall system performance. Efficient resource allocation is still a problem among D2D user equipment. This thesis proposed an idea for resource allocation by using deep reinforcement learning. This thesis considers improving power allocation to the devices by considering the interference caused by D2D pair, cellular and base station. This interference may degrade the power allocation of the devices. Using deep reinforcement learning, the resource allocation can be improved which will significantly improve the overall throughput of the system. The idea is to implement a decentralized resource allocation mechanism for D2D communication based on deep reinforcement learning to control spectrum transmission and distribution power to maximize system throughput by limiting minimum signal to interference noise ratio (SINR)on both cellular and D2D link with leverage, where each D2D connection is considered an agent that makes it decision to find optimal spectrum and power transmission. Finally, we compare our work with existing reinforcement learning based method. The result from the simulation shows that our proposed resource allocation algorithm outperforms existing methods regarding overall system throughput and D2D throughput too.

Annotatsioon
Dektsentraliseeritud Ressurside Jaotamine Side Seadmekülgimiseks

Kasvavat nõudlust kohalike teenuste, seadme vahelise (D2D) kommunikatsiooni järele peetakse järgmise põlvkonna kärgvõrkude oluliseks komponendiks, et parandada spektri taaskasutamist, suurendada ribalaiust ja süsteemi läbilaskevõimet. See võimaldab kasutajate seadmetel suhelda teiste UE-dega otse. D2D-ühendust mobiilsidevõrgus määratletakse kui otsest suhtlust kahe mobiilikasutaja vahel ilma tugijaama ületamata. Lähedal asuvate mobiilseadmete vaheline otseühendus parandab spektri kasutamist ja üldist läbilaskevõimet. Üldise süsteemi jõudlus võib aga halveneda, kui D2D kasutajate tekitatud häirete üle puudub õige kontroll. D2D kasutajaseadmete seas on endiselt probleemiks ressursside tõhus eraldamine. See lõputöö pakkus välja idee ressursside jaotamiseks sügava tugevdamise õppimise abil. Selles lõputöös kaalutakse seadmete energiajaotuse parandamist, võttes arvesse D2D paari, raku ja tugijaama põhjustatud häireid. Need häired võivad halvendada seadmete energia jaotust. Põhjaliku tugevdamise õppimise abil saab ressursside jaotust parandada, mis parandab oluliselt süsteemi üldist läbilaskevõimet. Idee on rakendada detsentraliseeritud ressursside jaotamise mehhanism D2D-kommunikatsiooni jaoks, mis põhineb sügaval tugevdamise õppimisel, et reguleerida spektri edastus- ja jaotusvõimsust, et maksimeerida süsteemi läbilaskevõimet, piirates signaali ja nina minimaalset suhet (SINR) F nii raku- kui ka D2D-lingil koos võimendusega, kus iga D2D ühendust peetakse agendiks, mis otsustab leida optimaalse spektri ja jõuülekande. Lõpuks võrdleme oma tööd olemasoleva tugevdusõppega. Simulatsiooni tulemus näitab, et pakutud ressursside jaotamise algoritm edestab olemasolevat meetodit nii süsteemi üldise läbilaskevõime kui ka D2D läbilaskevõime osas.

List of abbreviations and terms

DPI	*Dots per inch*
TUT	Tallinn University of Technology
D2D	Device to Device
SINR	Signal to Interference noise ratio
RB	Resource block
UE	User equipment
CU	Cellular user
DQN	Deep Q-learning

Table of contents

Contents

List of Figures

1 Introduction
1.1 Background

Communication between devices (D2D) in mobile networks is a direct connection between two mobile users without going through a base station or core network. D2D communication is usually opaque to the cellular network and can occur over cellular frequencies (i.e., within the frequency band) or unlicensed spectrum (i.e., over the frequency band).

In conventional mobile networks, all communications have to be passed through base station, either communication parties are under the range for a device-to-device proximity connection. Communication via BS is compatible with conventional low-data-rate mobile services, such as voice and text messages, where users are rarely close enough to communicate directly. However, today's mobile network users use high-speed data services (for example, video sharing, gaming, and proximity communication networks) where they may be in the range of direct communication (i.e. D2D). Therefore, D2D communication in such scenarios can significantly increase the efficiency of the network spectrum. The benefits of D2D communications exceed spectrum efficiency; they have the potential to improve capacity, energy efficiency [1] [2]

Communication between devices is a unique feature for the long-term development of modern systems that can operate in a centralized mode, that is, in a controlled and coordinated mode of a base station, and in a non-centralized mode, that is, without control and coordination of the base station. In D2D communication, it permits direct communication between users, reusing spectrum resources, which provides higher system throughput and D2D throughput. However, D2D devices interfere with the reuse of spectrum resources. D2D stated as a technique through which devices can communicate with each other without any base station or infrastructure of access points. D2D communication is a technological component for LTE-A, where a user equipment (UE) transmits data signals to each other on a forward link / connection using cellular resources rather than an eNB (i.e., base station).

As per the growing demand for local services, device-to-device communication is considered as an integral part for next-generation cellular networks to improve frequency reuse, throughput, better battery lifetime of users equipment by reusing the spectrum resources, leap forward and increase system capacity.

1.2 Problem Description

In D2D communication system, the D2D UEs can choose to communicate via serving BSs or transmit data via mobile links through direct links to other devices. Choosing a mobile or direct D2D connection fixes the problem of mode selection. Before selecting a device mode, other neighbouring devices must be found using the peer discovery procedure. After selecting a mode, the system must appropriately allocate spectrum resources to the cellular and D2D links, monitor transmit power, and control interference between the D2D link and the cellular structure. With knowledge of current network load, channel conditions, and potential D2D pairs, the system can select the best mode, allocate resources, and efficiently manage interference to realize the proximity, reuse, and hop amplification (gain) provided by the D2D connection. Thus, mutual recognition, resource allocation mode, and interference management are major technical challenges in advancing the cellular system underlying D2D communications. These problems are complex because they are related to device-level behaviour, interference-related situations at the cellular level, load at the network level, and channel condition.

Communication between devices offers lessons for the reuse of spectral resources. To achieve overall high system throughput, efficient resource allocation (which is power allocation in this case) plays vital role. If there is proper control on managing the power, D2d throughput will get better but it will affect the interference as well. Increase in transmit power can improve the D2D throughput but at the same time it increases interference level. Thus, in D2D communication, an important research issue is the selection of the correct level of transmission power.

D2D transmitters must take action at any time upon application request. For example, operations can be considered as various options for choosing a power level for a particular resource block. Planning these operations will help you increase system throughput and interference. Therefore, we need an algorithm to find out the suitable energy distribution operations in such a way as to provide better overall system efficiency and minimal interference.

Typical random distribution is not useful for a dynamic D2D communication scenario. Machine learning based algorithms improves overall quality of communication via training the operations for adequate level of power transmission allocation.

1.3 Challenges

Signal quality and interference are dynamic in nature because of which Random power distribution is not good enough for D2D connection. Adaptive power levels distribution helps to allocate resources for higher overall system throughput and to maintain acceptable interference levels. However, maintaining is difficult, and therefore our challenge to propose a method for adaptive learning , that will helps in improving the overall system throughput with a minimum interference.

The preliminary work on the distribution of adaptive power D2D was based on machine learning algorithms [3]. However, they experienced many challenges, for instance set of states and actions not considered properly. The reward function includes, as a limitation, only a signal to interference and noise ratio meter (SINR).They did not consider the channel gain between user and base station and among users too. In addition , the existing work on resource allocation does not provide an idea of the efficient use of resources. Moreover, there is always a scope to improve the system throughput using efficient method. Hence, the challenge is to adaptively select the transmit power level of D2D transmitters in a way that provides the higher system throughput and minimize the interferences.

Decentralized resource allocation in D2D communications is also a challenge. Centralized infrastructure requires full knowledge of the link state information that produces redundant information over the network.

1.4 Goals

The main goal of the thesis is to optimize the overall system throughput by considering the power allocation and limiting the aggregated interference by a predefined threshold. However, the transmitter should take one of Resource block (RB) where each can select one particular power level of each RB. Our aim here is to find the optimal resource allocation in order that the system throughput is optimized by applying deep reinforcement learning. The agent for this system will be designed by a neural network, which will help to perform action according to the states, actions and reward function.

The aim is to propose a decentralized resource allocation mechanism for D2D communication based on deep reinforcement learning to control spectrum transmission and distribution power

to maximize system throughput by limiting minimum signal-to-noise ratio (SINR) on both cellular and D2D link , where each D2D connection is considered an agent that makes its decisions to find the optimal spectrum and power for transmission. Because the proposed method is decentralized, each agent does not have to require global information to make his decisions, and the overall transmission costs might get low and each agent can learn how to meet D2D constraints while minimizing communication interference.

Moreover, the objective is to improve the existing algorithm so that the proposed method outperforms in terms of overall system throughput.

1.5 Structure of thesis

In section 1, we describe what is device-to-device communication, a brief introduction, description about problems in D2D communication , challenges in preliminary works and goals.

The rest of the thesis is organized in the form of different sections where each section is concerned about a different phase of work. However, Section 2 describes the related work done for device-to-device communication. This is followed by the system model in Section 3 which covers the overall architecture of the proposed method for dynamic resource allocation in social aware device-to-device communication. In addition, a proposed method for resource allocation which is based on deep reinforcement learning is described in section 4. Moreover, Section 5 in this thesis paper represents performance evaluation of my proposed method in comparison with existing methods.

Section 6 concludes the paper with future work proposals and recommendations.

2 Related works

Recent advances in reinforcement learning (RL) create a wide range of applications for responsive applications. Resource allocation in D2D communication is such an application. Here, we first describe some classical approaches [4] [5], and then present RL- based resource allocation algorithms [5] [3].

Reference [3] proposed a Local water filling algorithm (LWFA) for resource allocation in D2D Communication to maximize the overall system throughput. LWFA did not consider the interference among D2D pairs. It examine the relationship between number of resource blocks exist per D2D pair and highest power constraint for each D2D pair to provide a better system throughput. However, LWFA is a costly algorithm. Currently there are not too many works related to machine learning for resource allocation in terms of device-to-device communication. In [4], author proposed Distributed Resource Allocation Enhancement (DIRL) that is an algorithm based on Q-learning and generally used to enhance the system performance as compared of random allocator. Moreover, algorithm used by the author is not properly designed for instance set of actions and states are not properly diagnosed. Author's algorithm contains signal-to-noise ratio and noise power. However author did not consider any channel gain. However it is drawback because it is important to consider the increase in channels, since they help the D2D connection with the best SINR levels and transmit power, which is reflected in the increased system capacity.

In [5] Yin et al. proposed a distributed resource allocation method that supports minimum speeds for users and D2D pairs. A game theory algorithm (GTA) has been proposed to minimize interference between D2D pairs. However, this technique gives less spectral efficiency. In [6] , the authors explicate resource allocation method while maintaining QoS of cellular users and D2D pairs for the improvement of system performance. A method is designed, based on the three stages: the first system performs admission control, and the second system distributes power for each pair of D2D and its potential cellular users (CU). For the Maximum Two-way Matching (MBS) scheme, it is recommended to select the appropriate CU partner for each D2D pair with maximum system throughput. Though, this work primarily focuses on choosing the appropriate CU for resource allocation when adaptive power allocation is not considered. In reference [7], authors proposed a method where author considered both D2D pairs and cellular users synchronized. It was a centralized approach to resolve the problem where author considered the increase in interference line from the D2D transmitter to the BS.

They formulate the problem of allocating radio resources for D2D communication in the form of mixed integer nonlinear programming (MINLP). In addition, to describe more about MINLP it is computationally hard to resolve. However, in this complete heuristic approach author did not included adaptive power management.

In [8], the objective is to improve the overall speed of the system for which a method has been proposed by the authors that involves selection of a General D2D link mode and resource allocation. In the process of maximization of bandwidth using SINR and power, a problem has been noticed along with both D2D channels and cellular users for which a portable communal coalition formation game (CFG) has been offered by the authors in the form of solution to the problems. However, the problem of adaptive energy distribution has been ignored in this study. Reference [9] proposed an interference region where mobile users and D2D users cannot exist at the same time , hence called as restricted interference Region (RIR). Thus, an interference management mechanism has been provided by the authors that increases D2D output over time as a result of the adjustment in the size of RIR.

In [10], authors did not work on adaptive resource allocation to maximize throughput instead considered range of cellular users to gain maximum system throughput. Although author's proposed method is excellent in case of system failures. Moreover, this work put some lights on regional monitoring of violations. They do not need adaptive resource allocation to maximize system throughput. A general limitation of the aforementioned works is that they are completely centralized, which requires full knowledge of channel status information, which creates unnecessary information on the network. In [11], a resource allocation method has been used having high problem solving efficiency where an area limited by adaptive interference for several D2D pairs have been formed in which the selected D2D pairs share resources resulting in the increment in the throughput/ output of the system. However, this method has also the same limitation as that of [8] as it has also ignored the problem of adaptive energy distribution for users. In [12], the authors proposed an optimization algorithm, based on two-phases for adaptive resource allocation, which provides the best system throughput results. They offer the computationally complex Lagrangia dual decomposition (LDD).

We propose an adaptive resource allocation using Deep Reinforcement Learning (DRL), taking into account the proximity factor, further improvement in reward function, and the best state, actions. The proposed method is to resolve power allocation in D2D communication. Moreover, this work helps on reducing interference level which will increase overall system

throughput. As per the related works we considered to review, we can say that they focused on system bandwidth mainly. However they also considered D2D. Moreover, none of the existing solutions for resource allocation take into account the interaction of the transmit power allocation, planning online operations or adaptive resource allocation.

3 System Model

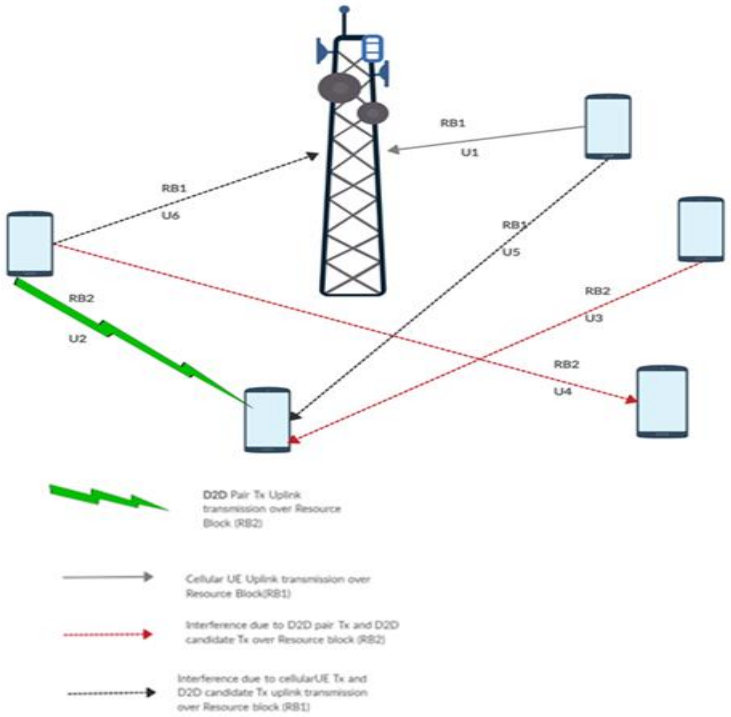

Figure 1 -D2D communication in cellular network

We consider cellular users (CU) and pairs of D2D users in one cell controlled by a base station. As shown in Figure 1, U1 denotes the CU uplink transmission on the resources RB1 , U2 denotes the pair of D2D Tx uplink transmission over RB2 whereas U3 is the interference due to D2D pair TX candidate on RB2. Similarly, the interference because of cellular users (CU) and candidates are U5 and U6 over RB1.

For the system model, let us consider a set of cellular users {1,2,3,....C} of C CU's and set of D D2D pair i.e. {1,2,3...D} with R number of resources block (RB). Each cellular user takes one resource block This resource block can be shared by D2D pair. The sharing of resource block may introduce several interferences:

- D2D transmitter can introduce interference in base station.
- Cellular users can introduce interference D2D receivers
- D2D can introduce interference in D2D receivers

If we find SINR for D2D users and cellular Users, it can help us to find link quality.

• For D2D users

SINR on the rth RB [13][6] :

$$\delta_r^{Du} = \frac{p_r^{DU}.G_{Du,r}^{uu}}{\sigma^2 + p_r^c.G_r^{cu} + \sum_{v \in D_r}^{v \neq u} p_r^{dv}.G_{Dv,r}^{uv}} \quad (1)$$

where p_r^{DU} represents the uth D2D UE uplink transmission power and p_r^{cu} shows cellular user uplink transmission power on the rth RB. $p_r^{DU} \leq P_{max}$, $\forall_u \in D$ whereas maximum of each D2D user`s transit power is denoted by P_{max}, $\sigma^2$is the noise variance, $G_{Du,r}^{uu}$, G_r^{cu}, $G_{Dv,r}^{uv}$are channel gains between uth D2D link, the channel gain between cellular transmitter (c) and receiver (u), channel gain for D2D transmitter (u) to receiver (v), respectively. $D_r$is D2D pairs set sharing the rth RB.

• For Cellular users

SINR on the rth RB [15] [8]:

$$\delta_r^C = \frac{p_r^{cu}.G_{c,r}}{\sigma^2 + \sum_{v \in D_r} p_r^{dv}.G_{v,r}} \quad (2)$$

similarly, $G_{c,r}$ indicate the channel gain on the rth RB between BS and cellular user c and $G_{v,r}$ indicate the channel gain on the rth RB between BS and vth D2D transmitter, respectively.

As there would be path loss as well which can be represented as:

$$\text{Total pathloss } \Delta L_{dB,B,u(.)} = L_{db(d)} + log_{10}(m) + log_{10}(X_u) - A_{dB}(\theta) \quad (3)$$

where as $L_{db(d)}$ shows the pathloss of BS which m meter far from user, X_u gives log normal shadow pathloss for user u and $A_{dB}(\theta)$denotes the radiation pattern[3][6].

where as,

$$L_{dB(d)} = 40(1 - 4 \times 10^{-3} h_{bs}) log_{10}(d/1000) - 18 log_{10} h_{bs}(h_{bs}) + 21 log_{10}(f_c) + 80$$

where $f_c$is the carrier frequency in GHz and $h_{bs}$is the base station antenna height.

For base station and UE communication, the linear gain is [15] [16]

$$G_{Bu} = 10^{\frac{-\Delta L_{dB,B,u}}{10}} \quad (4)$$

For D2D communication, the gain is [15] [17]

$$G_{uv} = k_{uv} d_{uv}^{-\alpha} \quad (5)$$

where as, $d_{uv}$is the distance between transmitter and receiver u and v respectively, α is constant path loss exponent and k_{uv} is normalization constant.

The aim of resource allocation is to allocate resources(i.e. resource block and transmit power) in in such a manner that helps to increase and improve the system throughput. System throughput can be said as summation of D2D users throughput and cellular user throughput.

Binary decision variable $a_v^{(r,\,p_r)}$ is used for resource allocation:

$$a_v^{(r,\,p_r)} = \begin{cases} 1, for\ the\ trasnsmitter\ which\ transmit\ over\ RB\ r\ with\ power\ level\ p_r \\ 0, otherwise \end{cases}$$

RB r can have total interference as:

$$I^{(r)} = \sum_{v=1}^{D} \quad \sum_{p_r}^{P_{max}} \quad a_v^{(r,\,p_r)} G_{v,r} p_r \quad (6)$$

Let A= $[a_1^{(1,1)}, \ldots\ldots a_1^{(r,\,p_r)}, \ldots\ldots a_1^{(R,\,P_{max})}]^T$ denotes resource(i.e. resource block and transmission power allocation)

Resource allocation can be given by [15]:

$$max_B \sum_{r=1}^{R} \sum_{p_r}^{P_{max}} a_v^{(r,\,p_r)} W_{RB} \quad \{log_2(1+\delta_r^C) + \sum_{u \in D_r} log_2(1+\delta_r^{Du}) \quad \} \quad (7)$$

subject to $I^{(r)} < I_{th}^{(r)}, \forall_r$

$$a_v^{(r,l)} \in \{0,1\}, \forall_{r,r,l}$$

$$\sum_{r=1}^{R} \sum_{p_{r=1}}^{P_{max}} a_v^{(r,l)} = 1, \forall_v$$

$0 \leq p_r \leq P_{max,} \forall_{u,r}$

where as p_r = ($p_r^1, p_r^2, \ldots\ldots p_r^R$), W_{RB} is for bandwidth of resource block RB. The goal of resource allocation is to allocate resources (i.e. resource block and transmit power) in such a manner that helps to increase and improve the system throughput. System throughput can be said as summation of D2D users throughput and cellular user throughput. Our goal here is to use Deep Reinforcement learning which will help in most advantageous resource allocation that can help to improve and maximize overall system throughput.

4 Proposed method for power allocation

4.1 Introduction

In this section we talk about how deep learning reinforcement will help out in resource allocation which is power allocation in our case. In reinforcement learning, we have states, action and reward. We have different states, according to the states we get the action and according to the action we set the reward, so that next time with that state it will take action with maximum rewards.

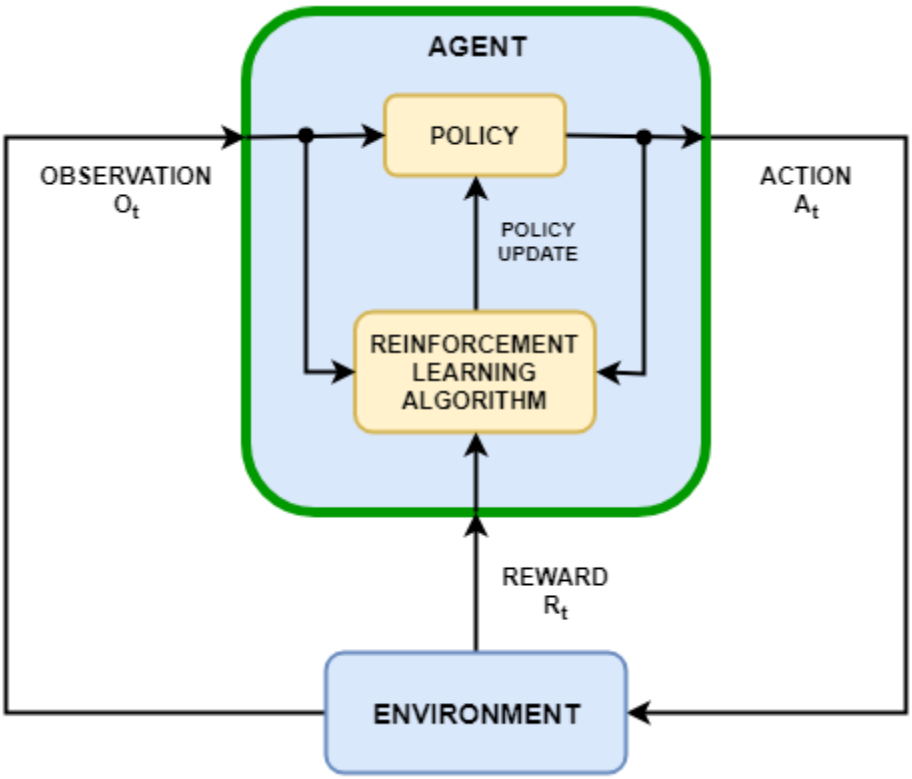

Figure 2 - Deep Reinforcement learning block diagram

Figure 2 shows the block diagram for deep reinforcement learning. We have observation (i.e. state), the agent which has basically the algorithm to get the action and the environment where we apply the action and set the rewards for the agent and select the next state for observation. Reinforcement learning is responsible for training the agent to achieve the task within an uncertain environment. Agent gets rewards and observation from the environment and sends action to the environment. Reward is set with respect to how successful an action was while achieving the task goal.

The agent consists of two components i.e. a policy and a learning algorithm.

- The policy is a mapping that selects actions based on environmental observations. Typically, a policy is a function equalizer with adjustable parameters, like a deep neural network.

- The learning algorithm constantly updates policy parameters based on actions, observations and benefits. Learning algorithm is the algorithm required to design the policy which will help in maximizing the accumulative rewards

Depending on the learning algorithm, the agent maintains one or more parameter function approximators for policy training. Function approximators can be classified as:

Critics: For this observation and action, the critic finds the expected value of the long-term future remuneration for the task.

Actors: For this observation, the actor finds an action that maximizes future long-term benefits [16].

Before talking about details on implementation, lets have an overview of machine learning and deep learning.

4.2 Machine Learning

Machine learning is a data analysis method that automates the creation of an analytical model. This is a discipline of synthetic intelligence based on the concept that structures can study from data, pick out patterns, and make selections with minimal human intervention. Machine Learning (ML) is the field of computer algorithms that are automatically learned and improved through experience. This is considered a subset of artificial intelligence. Machine learning algorithms create a mathematical model based on sample data called “learning data” to make predictions or make decisions without explicit programming [17]. Machine learning algorithms are widely used in various applications, together with e-mail filtering and computer vision, where it is hard or almost impossible to develop traditional algorithms to perform the specified tasks.

Machine learning is intensively related to computational statistics, which focuses on computer forecasting. The study of mathematical optimization provides different techniques, theories and

applications for the field of machine learning. Data mining is a related area of research focused on analysing the data being studied through teacher less learning [18]. Machine learning is also called predictive analytics in solving business problems.

The main goal of the learner is to summarize his experience [19] [20]. In this context, generalization refers to the ability of the learning machine to accurately perform new, invisible examples / tasks after the trial use of the training data set. The training examples are taken from some usually unknown probability distribution (which is considered representative of the performance space), and the learner must develop a general model for this space that allows him to make fairly accurate predictions in new cases. Computational analysis of machine learning algorithms and their performance is a section of theoretical informatics known as the theory of computational learning. Since learning sets are limited and the future is uncertain, learning theory usually does not guarantee the operation of algorithms. Instead, the likely boundaries of the show are pretty common. Deterioration of the bias variance is one way to quantify the generalization error. For best performance in the context of generalization, the complexity of the hypothesis should correspond to the complexity of the main function. If the hypothesis is less complex than the function, the model has sufficiently established the data. As the complexity of the model increases in response, the learning error decreases. However, if the hypothesis is too complex, the model may be appropriate, and the generalization will be worse. In addition to performance constraints, training theorists study the complexity and feasibility of learning. In the computational theory of learning, computation is considered possible if it can be performed during a polynomial. There are two types of results for time complexity. Positive results indicate that certain class functions can be studied during a polynomial. Negative results indicate that certain classes cannot be learned during the polynomial.

The learning process starts with monitoring or data, such as examples, experience or instructions, to look for patterns in the data and make appropriate decisions in the future based on the examples we provide. The main goal is to allow computers to learn automatically without human intervention and help and to adapt operations accordingly.

Machine learning is an artificial intelligence (AI) application that prepares any system to learn automatically while working from different combinations of input and output and improve from learning experience without explicit programming. Machine learning is proposed to develop computer applications that can be trained and learned from training. Machine learning

allows computers to solve tasks that so far only humans have performed. From driving a vehicle to speech translation, machine learning turns into a separate branch of artificial intelligence - this helps programs understand the messy and unpredictable real world.

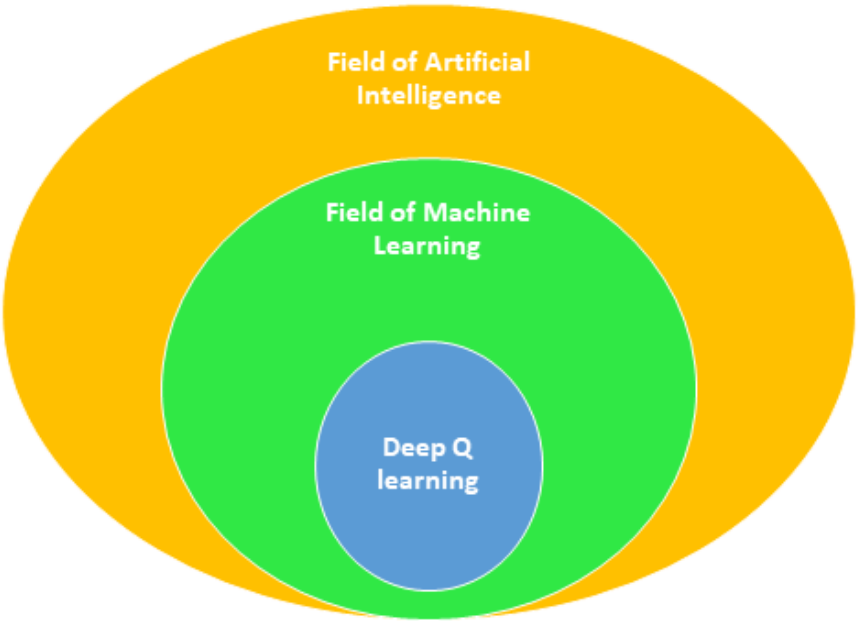

Figure 3 - Machine learning vs Deep Q-learning

At a very high level, machine learning is a process that teaches a computer system to make accurate predictions when it comes to data transfer. These predictions can be answered to the question whether the fruit in the photo is a banana or an apple, smearing people crossing the road in front of the self-propelled car, regardless of whether the dictionary is on paper or in a hotel, I have a video signature. The main difference from conventional computer software is that the developer did not write code to instruct the system on how to distinguish a banana from an apple. Instead, the machine learning model was taught how to reliably differentiate fruits by learning with a lot of data, in this case, probably, a huge number of images with a label containing a banana or an apple. Hence, data and many of them are the key to machine learning.

Due to modern computer technologies, machine learning today is not like machine learning in the past. In the beginning, ML was focused on pattern recognition and theory, which computers can learn without programming to perform certain tasks. Curiosity in the field of artificial intelligence was aimed to know if computers can learn from data. The iterative functions of machine learning is making it more important because models are uncovered to new data, they might adapt independently. They learn from previous calculations to get

reliable, repeatable solutions and results. This is a science that is not new, but has received a new impetus.

The types of machine learning algorithms vary depending on the approach, the type of input and output data, and the type of task or task that need to be solved.

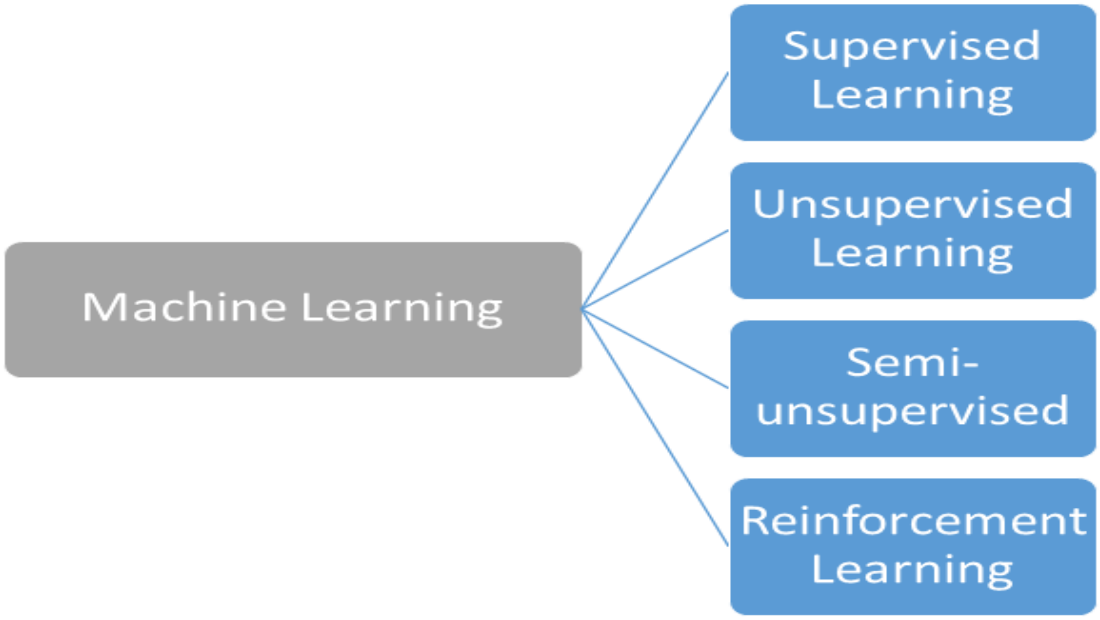

Figure 4 - Types of Machine learning

4.2.1 Reinforcement Learning

Reinforcement machine learning algorithms generally use learning methods that interact with your environment by performing actions and identifying errors or benefits. The most important features of reinforced learning are trial and error searching and late payment. This technique gives potential to software programs and machines to automatically determine the ideal behaviour in a specific context to maximize its performance. An agent needs simple feedback to find out which action is better. This is called a gain signal.

Reinforcement training is generally used for robotics, games, and navigation. Through reinforcement learning, the algorithm uses trial and error to determine which actions are most useful. This type of training consists of three main components: first the agent (decision maker), second is environment (everything the decision maker interacts with) and last is

actions (what the decision maker can do). The goal is for the agent to select actions that maximize the expected pay for a certain period of time. An agent achieves a goal much faster by following good policies. Therefore, the goal of on-the-job training is to learn the best policies.

To understand reinforcement training, you need to think about how someone can master the computer game of the old school for the first time if he is not familiar with the rules or does not know how to control the game. Despite the fact that they can be beginners, looking at the relationship between these buttons displayed on the screen and in-game results, their performance improves. An example of reinforcement training is the Google DeepMind Deep Q-network, which has beaten people up with a lot of old video games. The system assigns pixels for each game and determines various information about the status of the game, for example, the distance between objects on the screen. It then examines how the state of the game and the actions that it performs in the game are related to the result achieved. During many game cycles, the system ultimately builds a model that maximizes the result of the action in the maximum situation in which it is, for example, in the Breakthrough video game, where the paddle must be moved to listen.

Reinforcement is an area of machine learning that concerns how software agents should work in an environment in order to maximize total pay. Because of this generality, the field is studied in lots of different disciplines, like game theory and operations, even a few different fields as well along with research, control theory, information theory, simulation-based optimization, swarm intelligence, statistics, multi-agent systems and genetic algorithms. In machine learning, an environment is usually presented as a Markov decision-making process (MDP). Multiple reinforcement learning algorithms use dynamic programming techniques [21].

For reinforcement learning algorithms there is no need of exact knowledge about the mathematical model of MDP and it is generally used when exact models are not feasible. Reinforcement training algorithms are generally used in autonomous vehicles or when learning to play against a human opponent [22].

There can be three different techniques to implement a Reinforcement learning algorithm. There are described those three different techniques:

Value Based

Using a value-based learning method, you should try to maximize the value function of V (s). Using this method, the agent expects a long-term return of current countries as part of the π policy.

Policy Based

Using the policy-based RL method, you are trying to create a policy that allows you to receive the maximum rewards in the future in each state. Policy based techniques in reinforcement learning also have two different types of policy based methods such as deterministic approach and stochastic approach.

Model Based

In model based reinforcement training methods, you need to create a virtual model for each environment. An agent learns to perform in this particular environment

There are also defined some important characteristics of Reinforcement learning such as:

1. There is always a delay in feedback and not instantaneous.
2. Reinforcement learning follows sequential decision making.
3. Time is one of the crucial factors in Reinforcement learning based problems.
4. This type of learning does not have any supervisor, instead it has only a real number or a reward signal.

Learning Models of Reinforcement

In Reinforcement learning, mainly two models are used for learning process

1. Markov Decision process
2. Q-learning

Markov Decision Process

Markov Decision making (MDP) is a stochastic discrete-time manipulate process. It provides a mathematical basis for modelling decisions in situations where the results are partially random and partially controlled by the decision maker. MDPs are useful in investigating optimization problems solved by dynamic programming and reinforcement learning. TIRs were known at least in the 1950s. The core of Markov's research in decision making was Ronald Howard's 1960 book, Dynamic Programming and Markov Processes. During each time stamp, the process is in some state s, and the decision maker can choose any action available in the state s. The process responds at the next point in time, randomly moving to the new state s' and giving the decision maker the appropriate reward Ra (s, s ')[22].

The selected action affects the likelihood that the process will transition to the new state s'. In particular, this is provided by the state transition function P (s, s '). Thus, the next state s' depends on the current state of s and the decision to make a. However, given s and a, it is not necessarily dependent on all previous states and operations; in other words, TIR state transfers satisfy Markov's ownership[23].

There is list of parameters that used to get a solution with Markov Decision Process (MDP):

1. Set of Actions (A)
2. Set of States (S)
3. Reward (R)
4. Policy (n)
5. Value (V)

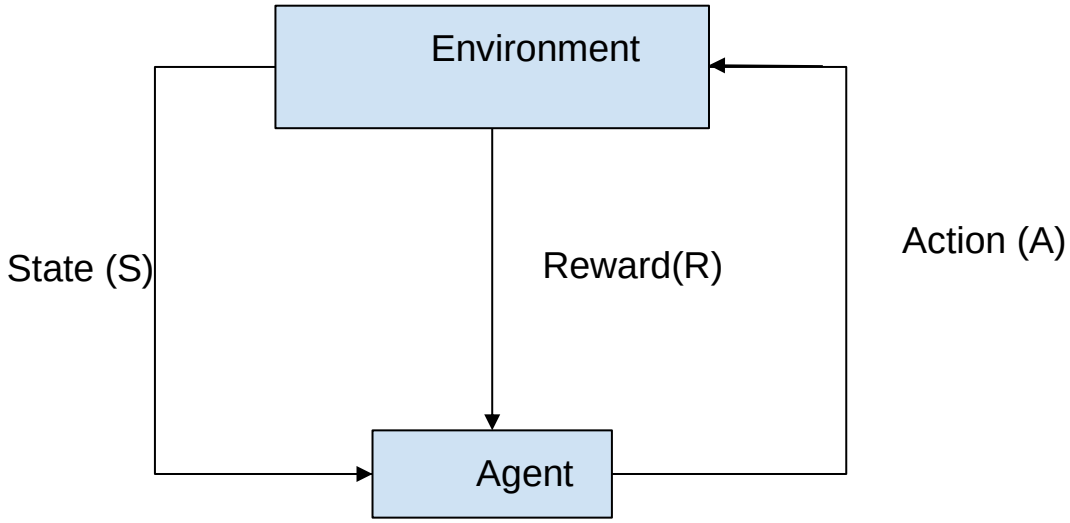

Figure 5- Block diagram Markov Decision process

Q learning

Q-learning is an algorithm for learning non-policy reinforcements, the purpose of which is to find the best actions in the current situation. This is not considered a policy because the Q-learning function learns from actions that are outside the current policy, such as performing random actions, and therefore this policy is not needed. In particular, q-learning seeks to study policies that maximize overall rewards.

Q-learning in reinforcement learning is based on value based technique to work with information or to inform which action should be taken by an agent.

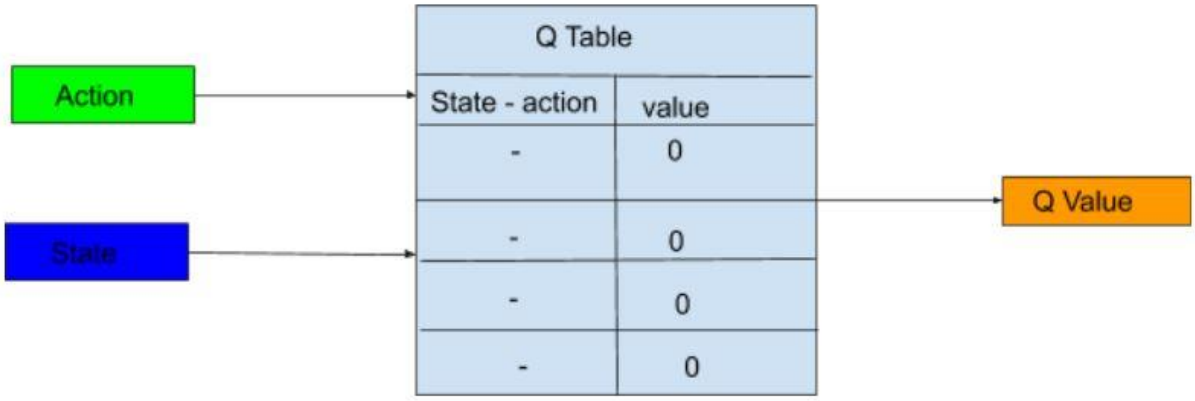

Figure 6 – Q-learning block Diagram

4.3 Deep Learning

Deep learning can be said as a subdomain of machine learning concerned with algorithms that imitates the working of the human brain for taking any decision. This algorithm is based on the function of the brain which is known as an artificial neural network. Deep learning method consists of neural networks that is why it is also referred to as deep neural networks. Now the question arises is what is neural network? In simple words, the neural network is a mathematical model of the brain which can be used to get the output from the input like human brain does.

A neural network could be a computing model whose layer resembles the networked structure of neurons within the brain, with layers of connected nodes. A neural network will learn from data—so it is trained to acknowledge patterns, classify data, and forecast future events.

A neural network breaks down the input into layers of abstraction. It may well be trained over several examples to acknowledge patterns in speech or images, for example, simply because the human brain does. Its behaviour is outlined by the method its individual components are connected and by the strength, or weights, of these connections. These weights are mechanically adjusted throughout coaching per a specified learning rule till the neural network performs the specified task correctly.

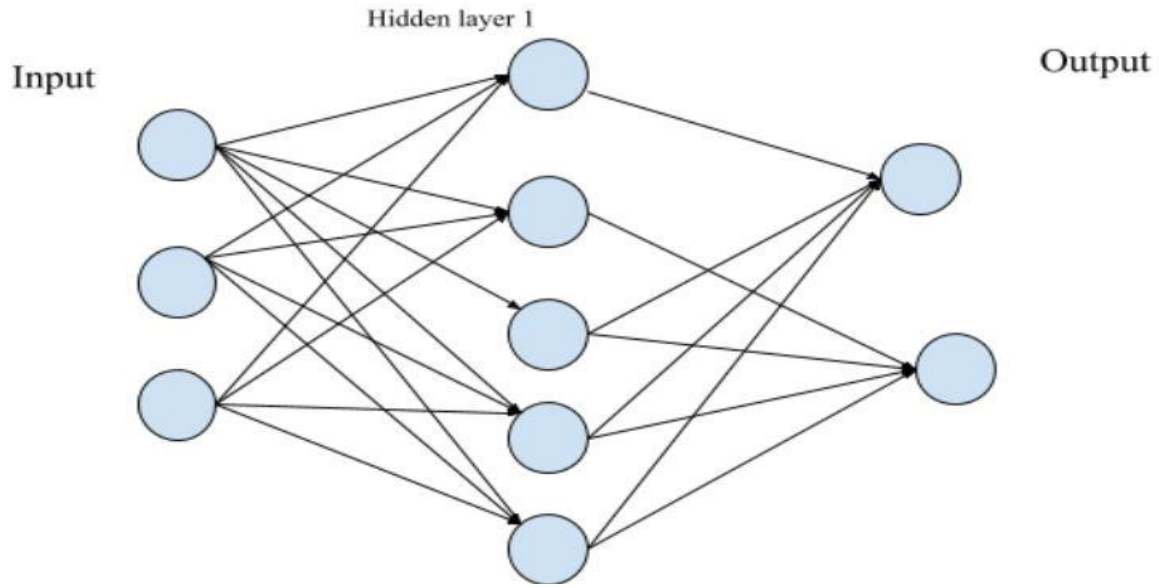

Figure 7- Neural network

There are 3 types of layer in neural network:

1. Input layer: This layer takes the initial data for the neural network.
2. Hidden layer: This is the intermediate layer which is responsible for computation.
3. Output layer: This layer gives the result for the inputs.

Let's break figure 7 as shown figure 8 to get the mathematical expression for node in hidden layer:

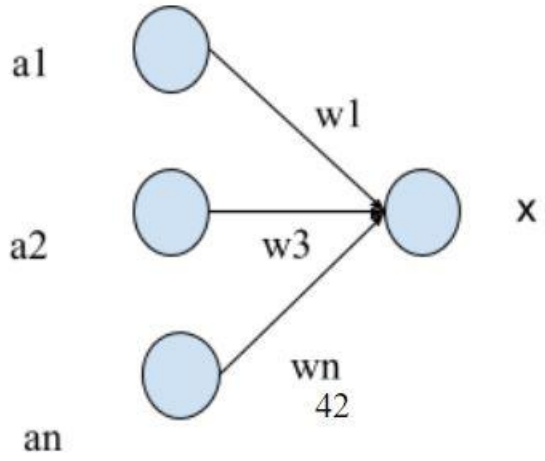

Figure 8 - Neural network with one activation node

w1,w2,...wn is the weight assigned between our neuron and neurons from the 1st layer. Now we take activation from the 1st layer and compute the weighted sum according to the weight.

$$w1.a1+w2.a2+.....wn.an \quad (8)$$

when we compute this sum we get some value and the value is passed to the needed activation function,

for example:

$$(w1.a1+w2.a2+.....wn.an) \quad (9)$$

we may need some bias, so will add bias to the weighted sum before plugging to activation function which gives:

$$x = (w1.a1+w2.a2+.....wn.an+b) \quad (10)$$

Bias function is used to shift the activation and helps in better prediction for the result

similarly it is calculated for all of the nodes in the next layer.

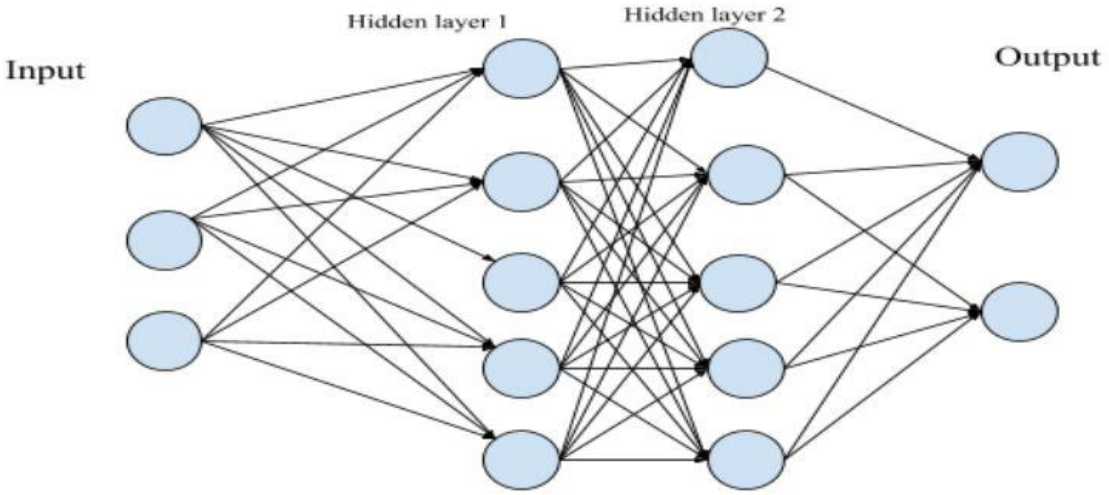

Figure 9 - Neural network with two hidden layers

Figure 9 shows that a massive and deep neural network has more layers and several more nodes in every layer, which leads to exponentially many more parameters to tune. We cannot tend to learn parameters efficiently without enough data. The learning would be slow without a powerful computer. The term deep represents the number of hidden layers in a neural network. It should have at least 2 hidden layer but it could have more than 100.[23]

4.4 Deep Reinforcement learning Implementation

We already discussed an overview of reinforcement learning and deep learning in section 4.1 and section 4.3.In Deep reinforcement learning, we use neural networks while designing the agent model which helps to perform the action of the task with observation rewards provided by the environment. The state for the agent is given by [24] [25]:

$$S_t^{u,r} = \delta_r^C \cup G_{Bu} G_{Uv} \quad (11)$$

where $S_t^{u,r}$ shows the state of D2D user u at RB r. In the equation of the state, there is 3 variables where $\delta_r^C$is the SINR of the cellular user at the rth RB, $G_{Bu}$is the channel gain between user u

and BS and $G_{uv}$is channel gain between D2D users u and v. The reason behind choosing these parameters is that SINR shows the quality of the network whereas channels gain influence in throughput because if the gain is good, we don't have to transmit more power but if the gain is bad we need to transmit more power which attracts interference. The value of each variable is changed in binary representation by setting threshold value for each.

For SINR δ_r^C,if the value is greater than τ_0 it is considered to be 1 and if less than τ_0 it is considered as 0. For G_{Bu} as well if the value is greater than τ_1 it is considered to be 1 and if less than τ_1 it is considered as 0. Similarly, for G_{uv} if the value is greater than τ_2 it is considered to be 1 and if less than τ_2 it is considered as 0

τ_0, τ_1, τ_2 is SINR and the channel gain

Action: Transmitting power level is the action performed by agent which is denoted by:

$$A=(A_r^1, A_r^2 \ldots, A_r^{pl})$$

here *r* is the rth resource block and *pl* is power level form agent.

Reward function: The reward function is set to maximize the throughput which is given by:

$$R=\log2(1+SINR(u)) \quad (12)$$

and penalty is given by

R=-1

4.4.1 Implementation using matlab

Matlab is one of the strong tools which has all built in software toolbox for built-in agents. The agent can be trained in environment which has continuous or discrete space with following action space:

Table 1- Built in Agents in Matlab

S.n.	Agent	Action
1	Q-Learning Agents	Discrete
2	SARSA Agents	Discrete
3	Deep Q-network Agents	Discrete
4	Policy Gradient Agents	Discrete or Continuous
5	Deep Deterministic Policy Gradient Agents	Continuous
6	Twin-Delayed Deep Deterministic Policy Gradient Agents	Continuous
7	Actor-Critic Agents	Discrete or Continuous
8	Proximal Policy Optimization Agents	Discrete or Continuous

Here, Deep Q-network has been used as an agent. The Deep Q-network (DQN) algorithm is a model-free, online, off-policy, value-based reinforcement learning method that trains a critic to estimate future rewards. DQN is a variant of Q-learning [16]. Table 1 show different built in agents for deep reinforcement learning in matlab, Here, Deep Q-network has been used as an agent. The Deep Q-network (DQN) algorithm uses model-free, online, off-policy, value-based reinforcement learning methodology that trains a critic to estimate future rewards [26].

As our observations have finite state and finite action, DQN can be used for our case. The agent of DQN can be trained in the environment with discrete observation and action space. While training the agent, the agent updates the critical properties during learning. It explores the action space with the help of epsilon-greedy exploration. During each control interval the agent selects a random action with probability ϵ, otherwise, it selects an action greedily for the value function with probability 1-ϵ. The value function is greatest for the greedy action. Then it stores the past occurrence in a circular experience buffer. The critic is updated based on a mini-batch of experience randomly sampled from the buffer.

To create the agent:

1. We need to have an environment which will provide reward and observation to the agent.

For creating custom environment in MATLAB:

```
env = rlFunctionEnv(obsInfo,actInfo,stepfcn,resetfcn)
```

It creates a reinforcement learning environment in which we have to provide the observation specification(obsInfo) and agent specification(actInfo). We need to have a matlab function to define our step and reset function. The reset function here sets the default state of the environment, this function is defined as myResetFunction.m. THe stepp function is used to specify how the environment will move to the next state according to the given action. The step function defined as myStepFunction.m.

We would have 8 state as we have 3 variables [25] and the 8 actions, so our environment can be designed as:

```
oinfo = rlNumericSpec([8 1]);
oinfo.Name = 'States';
oinfo.Description = 's1, s2, s3, s4, s5, s6, s7, s8';
ActionInfo = rlFiniteSetSpec([3 23]);
ActionInfo.Name = 'Action';
env =
```

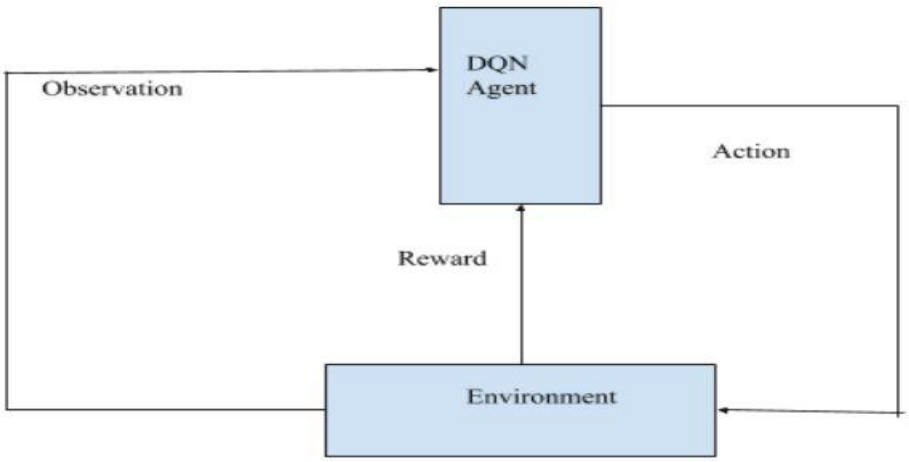

Figure 10- Block diagram for DQN

```
rlFunctionEnv(oinfo,ActionInfo,'myStepFunction','myResetFunction');
```

2. After getting the rewards and observations from the environment we need to design function approximators to estimate the value function. DQN agent have two function approximators to estimate the value function:

Critic $Q(S,A)$: The critic takes observation S and action A as inputs and outputs the corresponding expectation of the long-run reward.

Target critic $Q'(S,A)$: The target critic is periodically updated based on the critic parameter values to enhance the stability of the optimization

After the training, the critic $Q(S,A)$ is updated with a trained value function approximator [26].

According to the type of agent being used, MATLAB Toolbox software for reinforcement learnig have the following function approximators:

$V(S|\theta V)$: Critics estimate long term reward based in the observation

$Q(S,A|\theta Q)$ — Critics estimate long-term reward based on the observation S and action A

$Q(S,Ai|\theta Q)$ — Critics estimate long-term reward for all possible discrete actions given observation S

$\mu(S|\theta\mu)$ — Actors select an action based observation S

Actor and critic function approximators can be created using deep neural networks. We can use the MATLAB Deep Learning Toolbox feature for it. In this case we are concerned about critics.

The critic networks must match the corresponding action and observation specifications with the training environment. To obtain the action and observation dimensions for environment env, We can use the getActionInfo and getObservationInfo functions to get the action and observation dimension for environment(env).

```
actInfo = getActionInfo(env);
actDimensions = actInfo.Dimensions;
obsInfo = getObservationInfo(env);
obsDimensions = obsInfo.Dimensions;
```

In DQN agents the critic network take both observations and actions as inputs. Here, the dimensions of the input layers should match the dimensions of the corresponding environment observation and action specifications.

Deep neural networks have series of interconnected layers.

```
observationPath = [fullyConnectedLayer(8,'Name','CriticObsFC1')
    fullyConnectedLayer(8,'Name','CriticObsFC2')
    fullyConnectedLayer(8,'Name','CriticObsFC3')
    fullyConnectedLayer(8,'Name','CriticObsFC4')
    fullyConnectedLayer(8,'Name','CriticObsFC5')
    fullyConnectedLayer(8,'Name','CriticObsFC6')
    fullyConnectedLayer(8,'Name','CriticObsFC7')
    fullyConnectedLayer(8,'Name','CriticObsFC8')];


actionPath = [fullyConnectedLayer(8,'Name','CriticActFC1')
    fullyConnectedLayer(8,'Name','CriticActFC2')
    fullyConnectedLayer(8,'Name','CriticActFC3')
    fullyConnectedLayer(8,'Name','CriticActFC4')
    fullyConnectedLayer(8,'Name','CriticActFC5')
    fullyConnectedLayer(8,'Name','CriticActFC6')
    fullyConnectedLayer(8,'Name','CriticActFC7')
    fullyConnectedLayer(8,'Name','CriticActFC8')];
commonPath = [fullyConnectedLayer(1,'Name','output1')
    fullyConnectedLayer(1,'Name','output2')
    fullyConnectedLayer(1,'Name','output3')
    fullyConnectedLayer(1,'Name','output4')
    fullyConnectedLayer(1,'Name','output5')
    fullyConnectedLayer(1,'Name','output6')
    fullyConnectedLayer(1,'Name','output7')
```

```
    fullyConnectedLayer(1,'Name','output8')];
criticNetwork = layerGraph(observationPath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
```

Deciding about the number, type, and size of layers for your deep neural network illustration may be tough and is application dependent. However, the most crucial part for any function approximator is whether or not the function is ready to approximate the best policy or discounted worth function for the application; that is, whether it's layers that may properly learn the options of your observation, action, and reward signals.

rlQValueRepresentation object is used to create critic representation for deep neural network before that we need to specify the learning rate.

```
opt = rlRepresentationOptions('LearnRate',0.5);
```

pass the environment action and observation specifications to the rlQValueRepresentation object, and specify the names of the network layers to which the actions and observations are connected.

The environment action and observation specification are passed to *rlQValueRepresentation* object.

```
critic =
rlQValueRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'ob
servation'},'Action',{'action'},opt);
```

While designing deep neural network and the representation following approaches should be taken:

- At the beginning, use the smallest network but with a high learning rate. Train the primary network to check whether the agent converges quickly to poor policy or behave in a random manner. If either of those problem occur, resize the network by adding additional layers or more outputs on every layer. Find a network structure that is just huge enough and does not learn too fast, and shows signs of learning and improving after an initial training period.
- A low initial learning rate can allow you to ensure if the agent is on the proper track, and help to know that the network architecture is good enough for the problem. For

difficult problems, tuning parameters is far easier once you decide on a proper network

when configuring deep neural network representation with DQN agents they won't learn something. It slow throughout the first episodes, and that they generally show a dip in accumulative reward early within the training process. Eventually, they'll show signs of learning after the primary few thousand episodes.

After creating the critict representation DQN agent is created:

```
opt = rlRepresentationOptions('LearnRate',0.5);

critic =
rlQValueRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'ob
servation'},'Action',{'action'},opt);

agentOpt = rlDQNAgentOptions('DiscountFactor',0.95);

agent = rlDQNAgent(critic,agentOpt);
```

This is how deep reinforcement learning has been implemented here using DQN agent which help us to find the appropriate power and help in improving throughput.

5 Results and Discussions

We implement our proposed Deep Q-learning algorithm and compare it with existing Q-learning algorithm [24] .

For simulation, the parameters are considered as $P_{max} = 23\ dBm, RB = 30, CU = 30, D2D\ users = 12,\ D2D\ radius = 20m,\ pathloss\ parameters = 3.5, cell\ radius = 500m,\ \tau_0 = 0.004,\ \tau_1 = 0.2512,\ \tau_2 = 0.2512, I_{th}^{(r)} = 0.001, W_{RB} = 180KHz$, $\in_{max}$= 0.3, $\in_{min}$= 0.1, $k = 0.25, \rho = 1,\ \gamma = 0.9,\ \gamma_1 = 0.5,\ \lambda = 0.5$.

In this thesis, I consider a constraint of resources which is with 30 resource blocks only. There are 12 device-to-device user pairs and 30 cellular users. I consider a cell with the radius of 500 m. There 12 D2D pairs and 30 cellular users are distributed commonly within the coverage range of base stations.

I consider three different constraints those are using to define states for quality of the service. Those constraints are :

τ_0 = 0.004 , τ_1= 0.2512 and third is τ_2= 0.2512.

In Deep Q-learning algorithm, I used different parameter such as discount factor $\lambda = 0.5$, $\in_{max}$ = 0.3 and $\in_{min}$ = 0.1. Hence, this was about simulation parameters those are described further I compared result with existing deep learning algorithm where Deep Q-learning have overall better performance.

5.1 Evaluation of System throughput

System throughput has been evaluated by varying:

1. Number of iterations of learning algorithms
2. Number of users
3. Transmit power

5.1.1 Comparison with respect to number of iterations

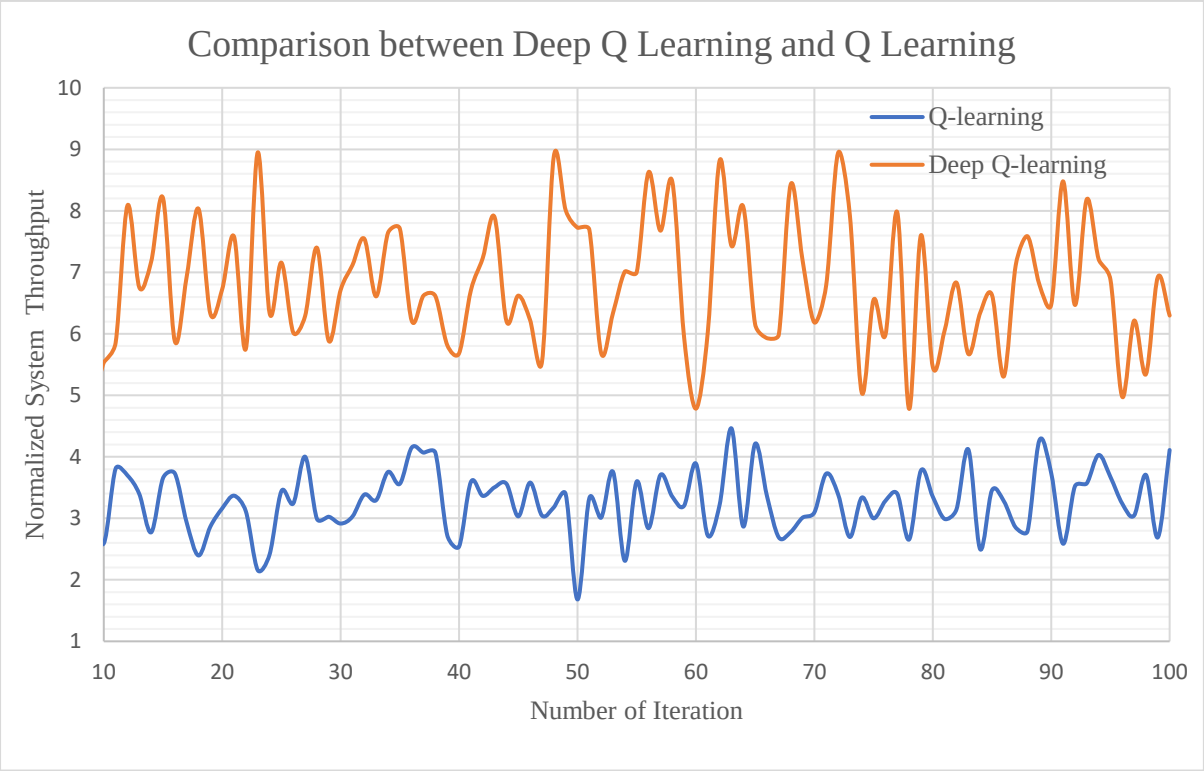

Figure 11- Comparison between Deep Q-learning and Q-learning

Figure 11 shows system throughput after applying Q-learning and Deep Q-learning. As we can see Deep Q-learning shows overall high throughput than Q-learning. I consider overall 100 iterations to examine the overall performance of both learning algorithms. We can observe that there are variations in result for both learning algorithms because of exploration and exploitation which is coming from action applied. In addition, we can also observe that there are variations in the system throughput for both methods, which basically for getting converged for the learning algorithms. We can observe that there are higher peaks for Deep Q-learning comparing with the Q-learning as Deep Q-learning consists of neural networks for training, which is more refined in terms of learning the actions to perform at each state. From the simulation, we can see that the overall performance of Deep Q-learning is better than Q -learning.

5.1.2 Comparison with respect to number of users

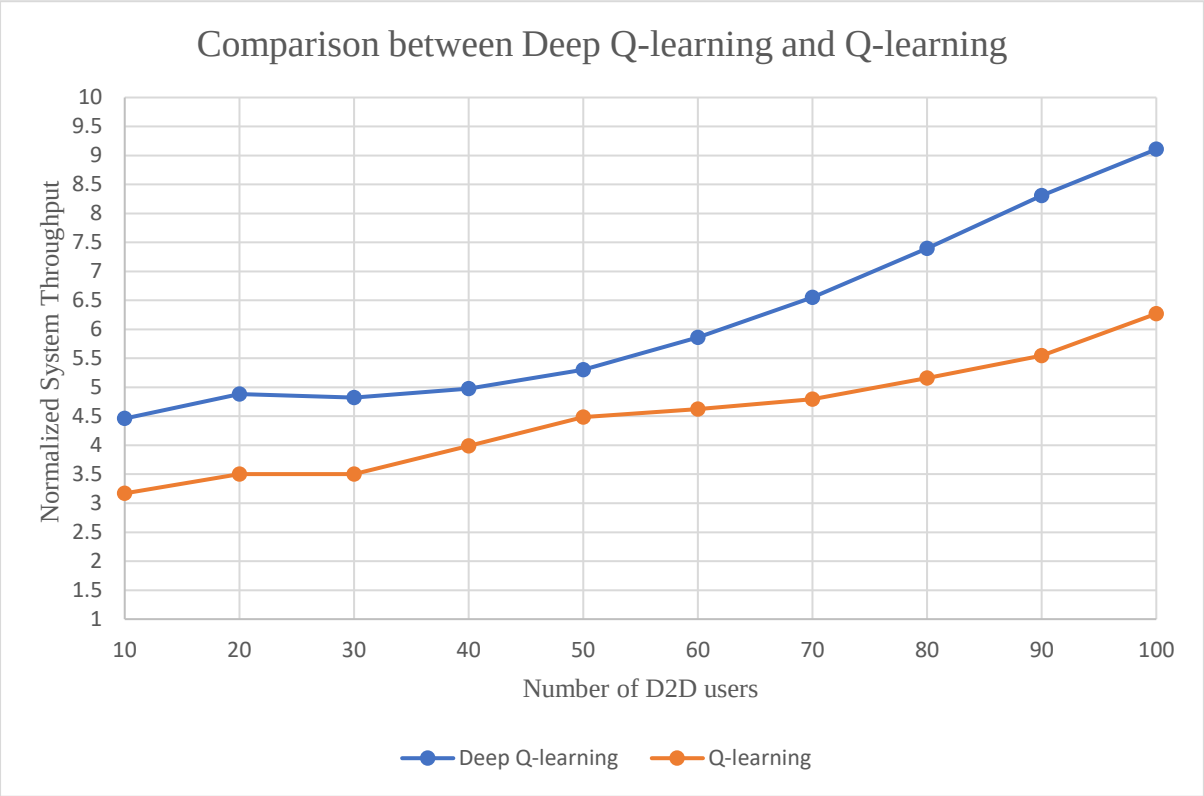

Figure 12 - System throughput over number of D2D users

Figure 12 shows comparison between Deep Q-learning and Q-learning with respect to number of D2D users. Here, system throughput is summations of D2D and cellular user throughput. We can observe from Figure 12, Deep Q-learning and Q-learning both are following the same trend with numbers of D2D users. In addition, Deep Q-learning gives higher system throughput as compare to Q learning. Overall, system throughput increases with number of D2D users. Although both algorithms perform well but Deep Q-learning gives outstanding performance over Q-learning.

5.1.3 Comparison with respect to transmit power

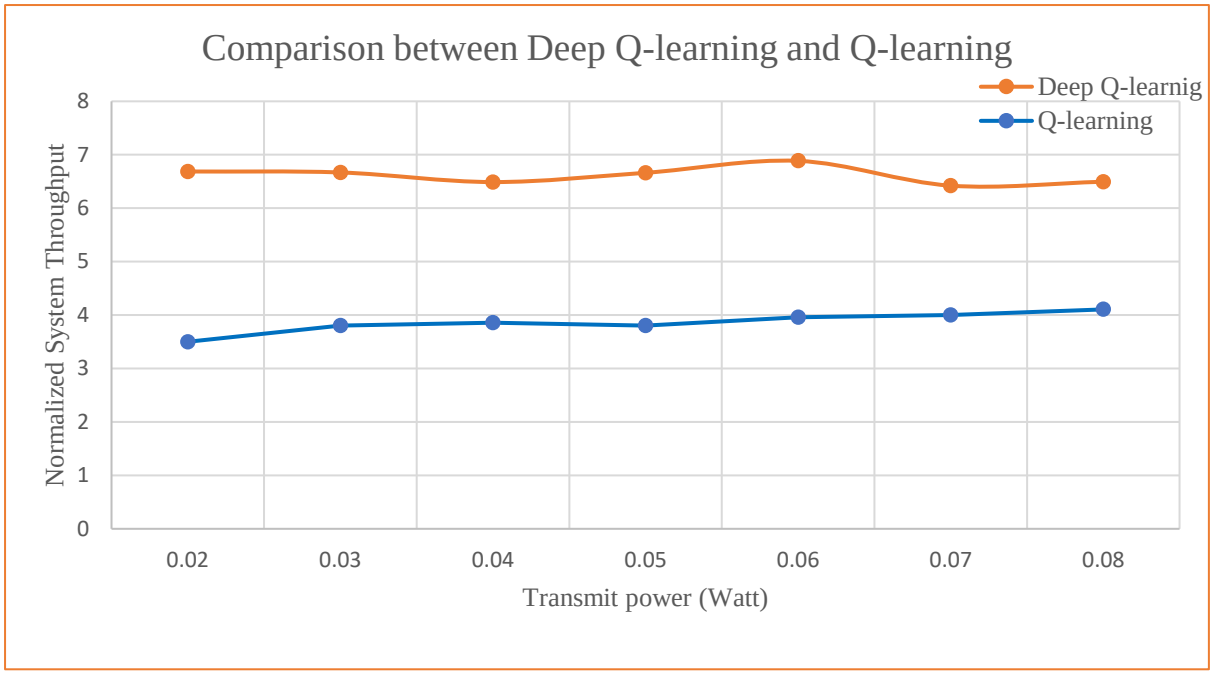

Figure 13- System throughput over transmit power

Figure 13 shows the average D2D throughput with respect to transmit power. Here, I have varied transmit power in my proposed method Deep Q-learning and compared it with the existing solution Q-learning. With the increment of the transmit power level, we can observe from the Figure 13 that Deep Q-learning shows overall higher system throughput as compare to Q-learning. At transmit power level 0.06 to 0.07, Deep Q-learning shows low performance comparing with the transmit power level 0.04 and 0.05 because Deep Q-learning requires higher amount of training time for better performance in long run. After the transmit power level 0.07, it increased. In addition, Overall, Deep Q-learning provides higher system throughput at every level of transmit power comparing with the Q -learning.

5.2Discussions

From above evaluation we can see that the overall throughput of the system gets better with Deep Q-learning. In figure 11,12,13, I have compared my simulation results with the Q-learning by varying number of iterations, D2D users and transmit power respectively. In figure 11, we can see that there is higher peak for Deep Q-learning than Q-learning and overall performance of Deep Q-learning is better than Q-learning. In figure 12, we can see that although there is increase in number of users, Deep Q-learning algorithm have outstanding

output. It depends on the training of the neural network. In figure 13, with increase of transmit power Deep Q-learning have higher system throughput.

6 Conclusion and Future works

Power allocation in D2D communication system is a critical issue. D2D cellular users can choose to communicate via serving BSs or transmit data or via mobile links through direct links to other devices while choosing a mobile or direct D2D connection fixes the problem of mode selection.

Our proposed method outperforms Q-learning in terms of system throughput. In this work, I consider more organized set of states and actions, gain between base stations and users and gain between 2 users. Results of this work shows outstanding performance for average D2D throughput, system throughput and throughput over number iterations. Proposed algorithm of Deep Q-learning performed overall better than Q-learning but still we can see variations in the result of Deep Q-learning itself which is due to exploration and exploitation. Due to this learning algorithm generally avoid to stuck on its local optimum value.

In addition, Deep Q-learning uses neural network for training, which helps to learn the right action to perform at right state and performs better over long run. However, it requires a huge amount of training time which is time consuming and can be considered as the drawback of Deep Q-learning in terms of time complexity.

In future work, we can compute the fairness index for proposed algorithm Deep Q-learning where fairness level of algorithm ensures its performance and also will calculate the time complexities.

References

[1] A. W. Q. M. Asadi, “A survey on Device-to-device communication in cellular networks.,” IEEE communications Surveys & Tutorials. , pp. 1801-1819, 2014.

[2] Y. E. Zhang, "Social Network Aware Device-to-Device communication in wireless networks.," in IEEE, 2015.

[3] S. Z. Z. Nie, “Q- learning based power control algorithm for D2D communication.,” in 27thIEEE Annual international symposium on personal, indoor and mobile radio communications, Valenica, Spain , 2016.

[4] Y. Z. H. Kai, “In proceedings of the Resource allocation for multiple-pair D2D communications in cellular networks.,” in IEEE International Conference on Communications, London, UK , 2015.

[5] R. Y. G. Z. H. L. G. Yin, “Pricing-based interference coordination for D2D communications in cellular networks.,” in IEEE Trans. Wirel communication. , 2015.

[6] D. L. L. Y.-W. G. Feng, “Device-to-device communications underlaying cellular networks.,” in IEEE Trans. commun, 2013.

[7] H. C. S. A. Zulhasnine, “Efficient resource allocation for device-to-device communication underlaying LTE network.,” in Piscataway, NG, USA 2010, Niagara Falls NU, Canada , 2010.

[8] J. c. K. C. Y. S. A.-Z. Zhao, “J. Joint mode selection and Resources allocation for machine-type D2D links.,” Trans. Emer, Telecomuno Technol., 2015.

[9] H. L. J. P. H. D. Min, “Capacity enhancement using an interference limited area for device-to-device uplink underlaying cellular netowrks,” IEEE Trans Wirel Commun , 2011.

[10] G. X. F. D. G. J. Yu, “Joint mode selection and resource allocation for device-to-device communication,” in IEEE Trans. Commun , 2014.

[11] R. S. J. Z. S. S. An, “Resource allocation scheme for device-to-device communication underlaying LTE downlink communications & signal processing,” in WCSP, Hunagshan China , 2012.

[12] H. E. M. I. Esmat, “Adaptive Resource sharing algorithm for Device-to-device communication underlaying cellular netowrks.,” IEEE, pp. 530-535, 2016.

[13] P. H. E. Z. K. Semasinghe, “An evolutionary game for distributed resources allocation in self-organizing small cells,” in IEEE, Mob. Compu. , 2015.

[14] F. S. F. Graziosi, “A gneral correlation model for shadow fading in mobile radio systems,” IEE commun. Lett. , pp. 102-104, 2002.

[15] H. C. S. Zulhasnine, “A Penalty function method for peer selection over wireless mesh network.,” in IEEE 72nd Vehicular Technology conference Fall , Ottawa ON, Canada, 2010.

[16] Mathworks, “Creating agents for reinforcement learning,” 2020. [Online]. Available: https://se.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html. [Accessed 2020].

[17] E. Alpaydin, “Introduction to machine learning 4th edition,” 2020, pp. 1-3, 13-18.

[18] J. H. Fredman, “Data Mining and Statistics: What's the connections?,” Computer science and statistics, pp. 3-9, 1998.

[19] C. G. C. C. s. Pavel Brazdil, “Applications to Data mining 4th edition,” in Springer science and Business media , 2009, pp. 10-14.

[20] U. o. Helsinki, “The Elements of AI,” Decemeber 2019. [Online]. [Accessed April 2020].

[21] B. C.M, Pattern Recognition and Machine learning, Springer, 2006.

[22] M. W. M. Van Otterlo, “Reinforcement learning and markov decision process,” in Reinforcement learning: Adaptive, learning, and optimization , 2012, pp. 3-42.

[23] M. B. C. Jordan, “Neural Network,” in Computer Science Handbook second edition , Chapman & Hall press , 2004.

[24] Khan, M.I., Alam, M.M., Moullec, Y.L. and Yaacoub, E., 2017. Throughput-aware cooperative reinforcement learning for adaptive resource allocation in device-to-device communication. Future Internet, 9(4), p.72.

[25] Kaelbling, L.P, Littman, M.L., Moore, “Reinforcement Learning : A survey”, 1996.

[26] Mathworks, “Create policy and value function respresentations.,” 2020. [Online]. Available: https://se.mathworks.com/help/reinforcement-learning/ug/create-policy-and-value-function-representations.html. [Accessed 1 April 2020].