IMPLEMENTING EFFICIENT PLANNING AND LEARNING ALGORITHMS FOR AGENTS IN MINECRAFT

Shalini Rajasingham

Registration No.148235B

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

IMPLEMENTING EFFICIENT PLANNING AND LEARNING ALGORITHMS FOR AGENTS IN MINECRAFT

Shalini Rajasingham

Registration No.148235B

Dissertation submitted in partial fulfillment of the requirements for the degree in Master of Science in Computer Science specializing in Data Science, Engineering and Analytics

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for Degree or Diploma in any other University or institute of higher learning. To the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: …….. Date: …….....

Name: Shalini Rajasingham

I certify that the declaration above by the candidate has carried out research for the Masters Dissertation under my supervision.

Signature of the Supervisor: …… Date: ……

Name: Dr. Surangika Ranathunga

ABSTRACT

An intelligent agent should possess the capability of solving problems related to the task of interest based on the perception of its virtual environment acquired from its past and present interactions. These agents should be able to extract the fundamental trait of being intelligent in order to possess human behavior. Learning and planning are the major modalities that contribute to this trait. Brown-UMBC Reinforcement Learning and Planning (BURLAP) is an existing library that comprises of algorithms that help the agent imitate the planning and learning behaviors of a human being. The algorithms in BURLAP can be used to implement intelligent agents in virtual worlds including Minecraft as it offers challenges that of a real-life platform. Minecraft allows the use of mods which are modifications to the environment based on the user's preference. The mod, BurlapCraft can be used to deploy the algorithms present in BURLAP. It includes scenarios such as dungeons that are of different caliber to test these algorithms. In literature, the developers of BurlapCraft have tested Rmax, Breadth First Search (BFS) and A star (A*) but have not implemented algorithms, Iterative Deepening A star (IDA star), Depth First Search (DFS), Q learning and State Action Reward State Action (SARSA) in BURLAP which makes the potential benefits of these algorithms unknown.

This research focuses on testing the efficiency and effectiveness of the reinforcement learning and planning algorithms, Q learning, SARSA, IDA star and DFS developed in BURLAP using the mod, BurlapCraft to make certain of their potential in solving a task oriented problem. It further analyses the potential of applying these algorithms in a pre-designed scenarios that are of different caliber which in turn would lead to the selection of the best fit and worse fit algorithms for the respective problems.

The performance evaluation identified that IDA star and Q learning algorithms do make an impact in improving the efficiency of the agent in completing the specified task. It also identified the best fit and the worst fit algorithms for the respective scenarios that could be mapped to general Artificial Intelligence (AI) related problems such as decision making, traversal and search present in the real world.

Keywords: BURLAP, BurlapCraft, Minecraft, Reinforcement Learning

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to Dr. Surangika Ranathunga, my supervisor, for her support and guidance in selecting and conducting this research. I would especially appreciate her patience and her feedback on the report, to correct, fine-tune and finally bring up to this level. I also appreciate our MSc coordinator, Dr. Amal Shehan Perera for his dedication and support. I would like to extend my gratitude to all the lecturers at the Faculty of Computer Science and Engineering, University of Moratuwa, for their valuable support. Further, I am grateful to my family especially my aunty, Ms. P. Anantheswary who has supported me throughout this effort. Finally I would like to thank my friends who have endured this long process with me.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation Description

BURLAP	Brown-UMBC Reinforcement Learning and Planning
VE	Virtual Environment
OOMDP	Object Oriented Markov Decision Process
MDP	Markov Decision Process
RL	Reinforcement Learning
AI	Artificial Intelligence
DFS	Depth First Search
BFS	Breadth First Search
IDA*	Iterative Deepening A star
SARSA	State Action Reward State Action

CHAPTER 1

INTRODUCTION

Background

Virtual Environment is an ingenious platform to explore the potential of intelligent agents. The versatility of the virtual environment and the human like nature of the intelligent agents create a replica of the real world which makes experimenting easier as performing robotics tests in the real world would be both expensive and unfeasible with respect to space and time. The intelligent agent in a virtual environment is based mainly on accepting percepts from the environment and generating actions that may or may not cater to the agent's goal. Such behavior can be manipulated using Artificial Intelligence (AI) techniques to solve problems similar to the ones encountered in the real world.

In literature [1][2][3] there exist countless aspects of what to expect from an intelligent agent. This report concentrates on two, learning and planning in a stochastic environment which play an important role in replicating the human behavior. Possessing such characteristics would enable the agent to be well equipped with diverse perceptual capabilities to face the challenges that are present in a complex environment. BURLAP, java code library assists in the development of single or multi-agent planning and learning algorithms along with the domains to accompany them. The structure of BURLAP is flexible for defining states and actions that supports discrete continuous and relational domains [4]. It is a versatile library that possesses algorithms that cater to the learning and planning behavior that range from classic forward search planning to value function based stochastic planning and learning algorithms. These algorithms present in BURLAP help the agents to develop learning and planning behavior that is similar to that of the humans.

Knowledge is consumed from nature by humans through interactions with each other, this in turn establishes a relationship in between them. It is through this bond that a community of rules, goals and goal directed actions are born. The experience through

these consequences of actions is how humans become efficient and effective in time. This learning process obtained from nature is transformed to a scientific approach called Reinforcement Learning (RL) which is focused more towards goal- directed learning from interactions [5]. There are two methods in how RL problems are tackled, one is model based and the other is model-free based [5]. The model based involves in solving RL related problems that use models and plans whereas model free based simply uses trial and error methods. Usually a model-based learning is considered to be a planning method and model free as learning.

The platform that is used to conduct the experiment is Minecraft [6], a java based virtual environment that also acts as a gaming platform. It is composed of three categories such as Blocks, Items and Entities. The intrinsic characteristics of Minecraft offer challenges that are similar to that of the real world. The complexity of the environment provides a simulation of the real world which makes it an excellent platform to experiment these learning and planning algorithms in BURLAP.

The complexity of Minecraft is tackled through organizing the raw data according to the relevance of the concept based on the situation. An agent's level of perception deteriorates with the complexity of the environment's state space, therefore this aspect compels the formation of a generalizing technique known as Object Oriented Markov Decision Process (OOMDP) [7]. OOMDP reflects the human cognition in interpreting the dynamic nature of the environment. Diuk et al. [7] believe that this representation structure incorporates the progressive nature of the environment into finite set of objects limiting the explosion of state spaces [7].

Minecraft extends the use of mods which are modifications made to the environment based on the user's preferences. The mod, BurlapCraft [8] is used to deploy the algorithms present in BURLAP. It includes scenarios such as dungeons that offer different levels of difficulty to test the learning and planning algorithms. In order to validate BurlapCraft as a platform for AI development the developers have tested a few algorithms, Rmax, BFS, A*[8].

1.1 Problem and Motivation

Current research [8] includes experiments on Rmax, A star and BFS algorithms hence the potential of other algorithms existing in BURLAP remains unknown. The problems existing in the AI world are complex and solving these problems demand the knowledge of other algorithms. This awareness would lead to selecting the best fit algorithms that are suitable for a given scenario that could be applied to the real-world problems. Even though it is difficult to find a tailor made fit for every algorithm, an estimated guess would still be a good start off to solve complex problems. Also, Minecraft caters to scenarios that are similar to that of the real-life problems hence it would be a feasible platform to test these algorithms.

1.2 Objectives

This research focuses on testing the efficiency of the reinforcement learning and planning algorithms, Q learning, State Action Reward State Action (SARSA), IDA* and DFS developed in BURLAP using the mod, BurlapCraft as the testing bed. It further analyses the potential of applying algorithms in pre-designed scenarios that are of different caliber which in turn would lead to the selection of the best fit algorithms that are suitable for a given scenario. These algorithms may later be applied to the problems in the real world that fall into the categories of such scenario to resolve the existing problem.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 discusses existing work related to Minecraft, OOMDP, generalizing technique, BURLAP, library that includes the learning and planning algorithms, BurlapCraft, the mod in which the learning and planning algorithms were deployed.

Chapter 3 presents the algorithms that to be tested in BURLAP in the dungeon created in BurlapCraft mod designed in Minecraft and evaluating the efficiency of the

algorithms based on the time it takes to generate an optimal policy.

Chapter 4 demonstrates the performance evaluation of the effectiveness of the work.

Chapter 5 summarizes the work and suggest future works.

CHAPTER 2

LITERATURE REVIEW

Virtual Environment (VE) provides a scalable platform in performing robotics tasks which is difficult to carry out in real life. The flexibility of the VE makes it an amiable tool and the complexity of the environment offers challenges that are faced in a real-life platform. Minecraft is a VE that comprise of a complex community, giving an opportunity to the researchers to test AI concepts with having the real-life scenarios in mind. These AI concepts are usually tested through intelligent agents that are present in the virtual environment and these agents usually possess human like qualities. Though there exist many qualities that researchers feel that is important for an agent to be intelligent, learning and planning are important traits that humans possess in order to be categorized as one. Similarly, these agents too must embed the learning and planning behavior in them in order to become or rather act intelligent. The compelling reasons to possess these qualities is for solving task oriented problems that are non-trivial especially when it is set in a complex environment.

BURLAP [4] is an existing library that comprise of learning and planning algorithms that helps the agent in Minecraft make decisions that produce outcomes that maximizes the reward function. The complexity of the environment is handled through OOMDP that is supported in BURLAP. Minecraft extends the use of mods which enables the users to customize the environment according to his/her preference. The mod that was used as the test bed to deploy the learning and planning algorithms is BurlapCraft [8]. The algorithms that are to be tested in the research is Q Learning, SARSA which is categorized under model-free learning and IDA* and DFS, categorized under a model-based learning approach.

This chapter gives an overview of the problem domain and then expands through the literature of the current affairs related to this research. It emphasizes the details of Intelligent Virtual Agent which are characters deployed in a Virtual Environment that is used to manipulate and execute tasks in Minecraft. The latter part of this chapter states the importance of using BURLAP and the relationship between Markov

Decision Process and OOMDP. Finally goes into detail of the algorithms related Reinforcement Learning and the impact it has on the learning and planning behavior.

2.1 Virtual Worlds

The evolution of Virtual World which was once within the boundaries of interactive games has branched over to various areas such as education, research, business and military. Virtual world is a 2– dimensional (2D) or 3–dimensional (3D) interactive computer simulated replica of the real world. It provides the user with an illusion in which the user is able to experience and manipulate the entities present in the modeled world. The user could take the form of an avatar and be able to interact with the fellow participants experiencing the virtual reality through sense and sound. These virtual worlds are designed to include a variety of functions such as exquisite visuals, role playing opportunities and animations that entice a wide range of target audience. Virtual worlds include, Disney's Virtual Magic Kingdom (VMK.com), General Mill's Millsberry, and Sulake Labs' Habbo Hotel [9], each of which are targeted to a specific age, demographics and functional applications while others such as Second Life, There, Active World and Minecraft [9] are targeted to a more general purposed audience.

Virtual World has been long since adopted in domains of massive multiplayer online games (MMOG). MMOGs have been further tagged under the terms, Multi-user Dungeon, domain or dimension (MUD), Collaborative Virtual Environment (CVE), Multi-user Virtual Environment (MUVE) and Massively Multi-User Online Role-Playing Games (MMORPGs). An exploratory factor analysis revealed a five factor model of user motivations - Achievement, Relationship, Immersion, Escapism and Manipulation - illustrating the multi-faceted appeal of these online environments [10]. MMORPGs are not only profitable platform for the entertainment sector but also a valuable research venue, hence virtual world offers an active economy that is designed around the ownership of virtual property and other forms of intellectual property as well [9]. Virtual Worlds like Second Life allow the user to indulge in their rich simulation and provides broader, network-based infrastructures, which allows the

researchers to examine the variety of economic organizational and social issues that extend beyond the domain of the individual user [9]. Though most of the virtual worlds are proprietary, there exists a few open source toolkits such as Open Simulators (OpenSim) which is similar to Second Life. The Linden Labs, owner and distributors of Second Life made the internals of the code belonging to the client software available to a broad array of users and developers, where a set of developers produced an alternate backend server that serviced this client which was possible since the client's functioning was made evident as part of the open source disclosure, this in turn resulted in the project called Open Simulator [11].

Although current virtual worlds have incorporated significant resemblance to the physical environment and communities, the human computer interaction has been improvised in the field of multimedia known as virtual reality where the humans can empathize with the communities through their senses. Technologies such as head-mounted display (HMD) and interaction device such as DataGloveTM aid in enhancing such interaction between the user and the virtual environment [12]. Thus, the combination of intelligent techniques and tools in the advanced computing and research communities are embodied in autonomous creatures and agents providing an effective means for graphical representation and interaction that has given rise to a new area known as intelligent virtual environments [13].

2.2 Minecraft

Minecraft is an indie game which was created in 2009 by Swedish programmer Markus "Notch" Persson and later published by the Swedish company Mojang. The game took a surprising leap in the gaming industry within a short period of time and currently having over 18 million registered users as of January 2015 [14]. The intent of the game is for the players to build constructions out of a textured cube in a 3D procedurally generated world. It also includes other functionalities such as exploration, gathering resources, crafting and combatting. It supports multiple gameplay modes including survival modes where the player must be able to acquire resources and to maintain health, creative modes where the player has unlimited resources to build and

to fly and adventure mode where the players play custom maps created by other participants [14].

The virtual environment of Minecraft is composed of three categories such as Blocks, Items and Entities. The block is composed of cubes, positioned in a fix grid pattern, representing various structures like water, dirt and stone. The entity is of anything that is capable of moving and that is affected by the gravitational force. This includes creatures, dropped items, arrows in midair, mine-carts, falling sand, players and more [6]. The player represents the user who interacts with the environment through survival and creation. The default physical appearance of this character is generally 1.79 meters tall, has indigo eyes, light brown skin, and dark brown hair. The final layer comprises of items that is usually handled by the player in order to manipulate the environment such as building a house, gather materials from the surrounding, chopping the trees down for wood, chipping away the cave wall for stone or digging the ground for dirt or sand. Minecraft is a sandbox game and hence the players are not required to accomplish specific goals as such and therefore promoting the user to be independent of choosing his own actions

Minecraft is a java based game that is not an open source. The users can make external modifications to the source code of Minecraft as long as no charges are made for their modifications. Mojang has not released an official API for Minecraft and hence unofficial APIs such as Forge and Sponge are available that enables the user to create mod files without making changes to the Minecraft source code itself [6].

The intrinsic characteristics of Minecraft make it a very suitable challenge for the state-of –the-art artificial intelligence techniques and hence broaden the horizon for goals that an intelligent agent could pursue which in turn helps in providing an excellent tool to test AI and MAS theories, applications and intelligent agent technologies in particular. In other words, experiments conducted in Minecraft are fairly inexpensive when compared to experimenting them in the actual scenario (real world) as trial and error testing and reconstructing the domain would not be feasible. The complex community of Minecraft gives an opportunity to the researchers to test the artificial intelligence concepts with having the real-world scenarios in mind. So that later these

algorithms could be fed into the actual robots to solve problems in the real world.

2.3 Intelligent Virtual Agents

The intelligent agents that are present in the virtual environment, which possess human like qualities and interact with humans, each other and their virtual environment is known as Intelligent Virtual Agent (IVA). IVA is a promising area of research that has emerged [15]. The notion of non-human agencies has been fascinating ever since history has been recorded [1]. The term agent in IVA necessitates the understanding of the context in which it is being used as Nwana [1] [16] and Ndumu [16] states that agents show a variety of appearances, perform a multitude of tasks and their abilities vary significantly which makes it challenging to produce a universally accepted definition. Shoham [17] defines an agent to be a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes. Detlor [17] describes agents as a newer class of software that acts on behalf of users to find and filter information, negotiate for services, automate complex tasks and collaborate with other agents to solve complex problems. Wooldridge and Jennings [18] on the other hand have introduced the weak notion of an agent in which it requires the agent to exhibit at least four of the following types of behavior: Autonomous behavior, Responsive behavior, Pro-active behavior and Social behavior. Although not stated explicitly, the communal notion of an agent that resonates in the above definitions is autonomy, though other characters such as, collaboration, solving complex problems, continuity are also included. Agents in general abide by concepts that emanate from an intelligent mortal. The challenges do arise in translating these psychophysiological expressions and mental states to a computational entity.

It is quite necessary to at least graze on the fields of AI and Multi Agent System (MAS) to compensate for the abstract notion of the term agent. The researchers Bates Maes, Shoham [2] working on AI conceptualize agent to be a computer system that in addition to the properties mentioned above implement theories that are quite applicable

to humans. These researchers embed the traits present in humans into the agents. For example, Shoham [15] characterize agents using mentalist notions such as capabilities, belief, choices, and commitment. Maes [19] on the other hand use the physical attributes present in humans to give a graphical representation of the agents. AI has largely contributed to the creation of intelligence such the ability to learn, comprehend, plan and so on. The trends to socialize integrated intelligence in computer systems has led to the emergence of the new field known as Multi Agent Systems (MAS). The idea behind MAS is the employment of multiple agents that would interact and perform independent actions in order to accomplish the design objectives on behalf of the manoeuvre (user or system). Such candidates should possess the ability to coordinate, cooperate and negotiate with the other fellow agents in order to fulfil the tasks of a MAS. The similarity in theories of both AI and MAS may force one to dissolve the boundaries each hold as there exists a time where the field of MAS was considered to be a subfield of AI or vice a versa [20]. Despite the similarities, AI and MAS include distinguishable traits that cause each of it to be defined on its own. The study of intelligence in AI fails to provide a system that is capable of making independent decisions, it is the integration of the entities that is related to intelligence that aid in doing so. The ignition of agent technology may require the concepts of AI but the major composition of it includes standard computer science and software engineering. As Etzioni [20] states intelligent agents are ninety-nine percent computer science and one percent AI. The other aspect that AI lacks is the ability of being social which is one of the major factor that contributes to the existence of MAS. It is important for a human to be intelligent but holding on to that trait only doesn't complete the human representation. There is a reason for humans to be called social creatures, it is through this ability that we build communities, beliefs and so on. Hence the goal of multi agent systems' research is to find methods that allow to build complex systems composed of autonomous agents who while operating in local knowledge and possessing only on limited abilities are nonetheless capable of enacting the desired global behavior [21].

As the terms intelligence and agent have been analyzed, it is time to consider pairing these two terms and evaluating its definition. So how does intelligence relate to

agents? On which benchmark is the intelligence of an agent tested? There is no definite answer to any of these questions. Hence Wooldridge and Jennings [2] have drawn a list of capabilities that an intelligent agent may showcase. The first includes relativity in which an agent should be able to perceive the environment and react to the changes in the environment appropriately satisfying the design objectives. Though a balance in reactive behaviour and goal oriented behaviour may be hard to achieve. The other is the social ability of an agent. In order for an agent to be social it should acquire characters such as cooperation, negotiation and sharing. It is necessary for an agent to be social in order to accomplish a goal. An agent must be intelligent to be able to cope up with environments which are dynamic, volatile and uncertain. Hence an intelligent agent system is a computer system that is capable of flexible (responsive, proactive and social) autonomous action in order to meet its design objectives [19]. Hereafter the term agent denotes as an abbreviation for "intelligent agents".

In literature the term IVA is referred to as virtual agents [22], bots [20] [23], or avatars [24]. As for setting out a consensus boundary on IVA, the definition provided by the international conference on Intelligent Virtual Agents (2015) is used:

"Intelligent virtual agents (IVAs) are interactive characters that exhibit human-like qualities and communicate with humans or with each other using natural human modalities such as facial expressions, speech and gesture. They are capable of real-time perception, cognition and action that allows them to participate in dynamic social environments."

The autonomous control in IVA is facilitated once it adorns a realistic character. The control depends on the physical accuracy and structural complexity of the body which could be computationally expensive [25].

The underlying concept of an intelligent agent is that it should possess the capability of making decisions based on the perception of its virtual environment acquired from its past and present interactions. The perception and the interactions of the environment are usually achieved through sensors and effectors. The sensors can be of any element from eyes, ears and other organs and effectors too can range from hands, legs, mouth and other body parts [3].

Russel and Norvig [3] gives an overview on what should be expected from an intelligent agent. They perform an analysis on agents based on their designs, of which includes rational agents that is expected to be righteous which is later evaluated as the success rate of an agent.

An agent program is based mainly on accepting percept from the environment and generating actions that may or may not cater to the agent's goal. The percepts are stored in an internal data structure which will be updated each time a new percept arrives. However, it depends on the agent to build the percept sequence in the memory. It is not all precepts that are recorded as it may not be necessary or even feasible in a complex environment. An important point that must be taken into account is that the goal or the performance measure is not programmed into the agent, it would be factored externally in a way to judge the behaviour of the agent and also not having an explicit knowledge of the performance measure may produce a higher one.

In order to build a real-world program, the concept of percept to action mapping must be clearly illustrated, it requires the involvement of different types of agent. One of which includes simple reflex agents which uses condition action rules or situation action rules to perform an action that matches the current state. For example, if an agent is maneuvering a vehicle it should follow a set of road rules, for instance when the traffic signal turns red, the agent should stop the car, likewise when it turns green it should continue to drive. Therefore, an agent should maintain an internal state that includes a systematic update on the world around it along with the impact it might have on the state of the world.

The current state or the internal state would be of no use if the agent does not have a goal. The agent based on the percept should decide on the actions that has to be executed in order to achieve the desirable state or goal. This concept is known as searching and planning in the subfields of AI [3]. It is important to note a fundamental difference between the goal designed agent and the reflex agent, the former involves consideration of the future, the actions based on the precepts are fine tuned to cater towards the agent's goal. The latter includes a set of pre-programmed rules for every possible scenario thus no decision making is involved. In other words, the goal based

agents will automatically alter its actions to the change in a goal or condition but on a reflexive agent that same change must be reprogrammed to suit the new condition or goal.

The success rate or the performance measure is a major component that evaluates the happiness of an agent. According to the authors, an agent achieving the goal may not be scoring a higher degree of happiness as it involves rational decision making in choosing the suitable states that could achieve it. The decision also includes choosing of a reasonable goal when there are several or conflicting ones. These tasks are managed by the utility agent. This agent makes rational decisions through comparing the utilities achieved by different courses of actions in contrast to the goal based which chooses an action as soon as it satisfies the goal.

2.3.1 Environment Perception

In agent technology perception is the ingestion of raw data from environment that needs to be processed in order to have a stratospheric view of the situations that is occurring in a virtual environment. The major challenge lies in interpreting these raw data and making sense of the actual array of incidents. These may range from complex situations such as assessing the opponent's strength to simply detecting and labelling objects present in the virtual environment. Hence the agent has to be well equipped with diverse perceptual capabilities in order to face the challenges present in the environment. Though recent research [26] has focused on achieving these capabilities there still exists a gap in accomplishing human like manipulation in the context of perception [26]. Hence this compels the notion of perception to be clearly defined.

Immanuel Kant divided the human mind into two basic cognitive faculties, the faculty of understanding and sensibility [27]. In the context of agent technology, it is possible to assume faculty of sensibility to be categorized as low level perception and faculty of understanding as high level perception. According to Kant's [27] theory the faculty of sensibility gears towards accumulation of raw data from the various sensory modalities and faculty of understanding organizes these raw data into concepts

providing a panoramic view of the environment. This report mainly focuses on high level perception of an environment, making sense of the raw data into a more comprehensive view of the environment encompassing concepts and situations at a non-representational level [28].

According to Chalmers et al. [28], high level perception includes levels of processing that involves concepts which play an important role and these levels range from concrete to abstract. Concrete level comprises of recognition of objects, for example being able to distinguish between a wall and a door, abstract on the other hand is the ability to assess complex situations such as politics or righteousness in imposing death penalty. Perceptions may have been influenced by belief where one reacts to an incident based on prior knowledge, influenced by goal where one object can be perceived as an obstacle or an useful entity based on their goal, influenced by external context, deciding on an attire based on situations and there might be other scenarios where perception can be influenced [28]. High level perception involves in extracting the meaning out of a given scenario and through this influence the formation of a concept which later leads to a conceptually driven process [28].

The concept that is produced through the process lies heavily on the state of the perceiver and the situation hence it is impossible for a concept to have a definite representation of the situation [28]. The representation of a situation is considered as a process according to Clayton T. Morrison and Eric Dietrich [29], as the interaction in between high-level concepts and low level processes, high level concept is influenced by low level processes and the level of perception at the low level affects the high level concepts, through which the process of representation at a conceptual level is constructed [29].

2.3.2 Representation of Environment

Traditionally the researcher's hand coded their best possible representation structure based on the knowledge they have of the problem and later the data that is assumed to

be relevant is organized by a programmer and fed into the constructed representation structure. Therefore, this structure ignores the problem of high level perception [27]. Building representations of an environment is not a trivial task as it requires a level of human cognitive abilities. Especially when it comes to interpreting a scenario and extracting the appropriate meaning out of it.

The complexity of forming a representation lies in organizing the raw data according to the relevance of the concept based on the situation. An agent's level of perception deteriorates with the complexity of the environment's state space. According to Sýkora [30], in order for an agent to be sensitive and react efficiently it should be able to explore every possible state and calculate the possible outcomes in these states and then choose appropriate actions and states that would produce the best reward [30]. In order to solve the exponential growth of states, in mathematics there exists a framework known as Markov Decision Process (MDP) which is suitable for virtual environments that are stochastic in nature and hence this could be modelled into stochastic decision-making problem [31].

2.3.3 Markov Decision Process

Markov Decision Process is defined by (S, A, T, R,) for a state space S, the decision maker can choose an action in action space A that is available in state s, this process leads to the transition probability $T(s^1 |s, a)$ of state s^1 , reward function R(s). The assumption of the Markov Decision Process is that the transition and the cost depend on the current state itself and not the states preceding it as the actions will be chosen only from that particular attained state itself.

The four parameters of MDP could be mapped with the environment the decision maker (agent) navigates. The states are either finite or infinite and they play a key role in stochastic decision-making problem. They are basically the platform in which the states the agent navigates. For example, in Minecraft, the infrastructure of the environment is constructed by blocks, each block or a room in a building could be considered as a state. The next is the action, these are finite set of built-in methods that

the agent get to choose in each state. The transition probability is what depicts the dynamic nature of the world. It is here where the next state of possible consequence is realized. Therefore, each state s_t and for each possible action a_t, the probability of the next state will be s_{t+1}.

In a dynamic environment where there is an exponential growth of the states, the process of learning becomes extremely challenging. In order to shrink the states, a form of generalizing technique must be deduced. Diuk, Carlos, Andre Cohen, and Michael L. Littman [7] have proposed Object Oriented Markov Decision Process (OOMDP), a representation structure that generalizes objects and its interaction into general categories that help model the environment in large spaces [7].

2.3.4 Extension of MDP: Object Oriented Markov Decision Processes (Generalization)

OOMDP reflects the human cognition in interpreting the dynamic nature of the environment. The researchers, Diuk et al. [7] believe that this representation structure incorporates the progressive nature of the environment into finite set of objects limiting the explosion of state spaces [7].

The OOMDP represents the state of the MDP through objects and predicates. An OOMDP comprises of a set of objects, O = {o1 . . . oo}, class cj ε {o1 . . . oo}, and attributes Att(c) = {c.a1 . . . c.aa}. Each of the objects belong to the class and these class in turn have a set of attributes. Each of these attributes have a value domain, Dom(c,a) of possible values. It also has the option of using set of predicates P over classes in order to provide a high-level information about the MDP state.

A real-life example such as a Taxi domain that is defined by Dietterich [7], will be used to clearly map the OOMDP concepts [7].

The scenario of the Taxi domain that is stated by Diuk et al. [7] includes a taxi that has a task of picking up a passenger and dropping it off in the pre-designated set of locations (Y, G, R and B). The dropping off location is the goal of this scenario. The

set of action that the agent can choose from includes North, South, West, East, Pickup and Drop-off. The obstacle in this grid world is the wall and this is what that limits the taxi's traversal. The objects in the Taxi domain that maps with the representation of OOMDP are the Taxi, Passenger, Destination and Wall. The Taxi, Passenger and Destination have attributes x and y that defines the location in the grid. Passenger has a Boolean attribute of in-taxi which states if the passenger is in or not. Walls have attributes that state the position it is located in the grid.

In a regular MDP, a wall would be considered as an object of a particular location and when an agent encounters one of each in different locations it would assume it as a different object regardless of its alikeness but in OOMDP, a wall would be considered as an object of the same category regardless of the location. For example, a wall encountered in location (4, 5) and a wall encountered in location (3, 8) would be assumed as two different objects in MDP but in OOMDP it would be assumed as the same object and would label it the same as the other. The latter would basically have a similar assumption of a human being. This reduces the strain on the agent as well.

Transition dynamics is induced through the interactions of objects and the effect that is produced. This interaction of the two objects plus the internal states of the two objects creates an effect. An effect for every pair of objects $o_1 \in C_i$ and $o_2 \in Cj$ that participate in the interaction is determined by the internal states o_1.state and o_2.state, the action, the boolean function from the set of relation r (o_1, o_2). It is basically the change of values of object's attributes.

When considering the taxi scenario, it includes the following relations: $touch_N$ (o_1, o_2), $touch_S$ (o_1, o_2), $touch_E$ (o1, o2), $touch_W$ (o_1, o_2), and on (o_1, o_2), which defines if each of the object $o_2 \in C_j$ is one cell North, South, East or West of object $o_1 \in C_i$ or if both the objects are on the same cell.

When the $taxi_i \in$ Taxi observes the current state s which is being on the northern edge and tries to move North. From the current state s, the taxi extracts all the relation in between two objects and observes the value of the attribute that is assigned. If the established relation $touch_N$ ($taxi_i$, $wall_j$) is true then the effect of this particular relation would be no change, if the relation $\neg touch_N$ ($taxi_i$, $wall_j$) is true then it performs an

action North and the effect of this will be $taxi_{i.y} \leftarrow taxi_{i.y+1}$ which is moving forward to the square of the grid. Then the environment chooses the reward r from the transitioned state R(s,a) after which the agent is notified with the reward that is received.

Another domain that uses the OOMDP representation is Minecraft which is described in detail above. The environment includes room and block objects. These objects can be defined by their positions in the world. These too like the taxi domain uses propositional function that is used on the block and room objects. Propositions such as "blockInRoom" would return true if the block is in the room and false if the block is not in the room. In Minecraft, agents, block, special room and inventory are considered as objects O_i. These could include even wider range of objects by categorizing them into classes C depending on the scenario such as farm animals, university system and so on. An example state representation from a Minecraft dungeon includes a sub object roomorange in Object that has attributes roomxMax, roomxMin, roomZMax, roomZMin and roomColor. Likewise, a similar sub object with it attributes is included for other objects such as block, inventoryblock and agent [8]. Similar to the taxi domain, these too have effects for every action performed and if the effect produced is a duplicate of the observed effect then the current effect is eliminated from the demonstration sequence. Since the actions performed in Minecraft are performed through keyboard and mouse clicks, each of these must be mapped to the actions in the Minecraft and then later the transition of the state relevant to this particular action is observed.

2.4 Reinforcement Learning

Knowledge is consumed from nature by humans through interactions with each other, this in turn establishes a relationship in between them. It is through this bond that a community of rules, goals and goal directed actions are born. There is no doubt in stating that such interactions are crucial to human's upbringing and understanding of the environment. The experience through these consequences of actions is how humans becomes efficient and effective in time. This learning process obtained from nature is transformed to a scientific approach called Reinforcement Learning (RL)

which is focused more towards goal- directed learning from interactions.

This is a third machine learning paradigm along with supervised learning and unsupervised learning where the former involves in training of data based on supervised predefined labels and the latter through identifying the hidden similarities of the data and clustering them based on it. RL is trapped in between exploration and exploitation, in order for the agent to produce maximum reward it has to choose an action that it has tested and proved to be effective but on the other hand in order to discover the most profitable action it has to try out actions that was not tested before [5].

The reinforcement learning paradigm includes four other sub elements besides an agent and the environment, it includes a policy, a reward signal, value function and a model [5]. A policy is basically a set of predefined rules that are established in order for the agent to follow, in other words it's a mapping from a state that an agent has attained to the action it should precede with respect to the state it is in. It could be implemented using a function, lookup table or in some cases a set of solid computations. The reward signal is the goal the agent should achieve in a RL problem. For each state achieved the agent receives an immediate feedback in terms of a numerical value that determines an agent's progress in achieving the goals. The main aim for the agent is to maximize the reward function as this could be either positive or negative. An agent will not be able to change the process in which the reward function is computed, that is it can only affect the output by change of its actions and not the problem itself. For example, if an agent meets up with an obstacle and receives a negative reward function, it could change the reward function simply by taking another action and not by tampering the reward signal itself. The next element is the value function, it basically is the estimation of the sum of the rewards to be accomplished. The reward is an immediate output of the action taken by the agent, but the value function is the expected accumulation of these rewards that later contributes in achieving the goal. The value in fact is the observation of the states that is likely to be followed and the rewards produced in the respective states. For example, in comparing to real life scenario, consumption of sweet treats could be pleasurable in the short term but in the long run it would affect your health. Same principle could be applied to this

concept as well. Rewards are important as values don't exists if there are no rewards, but it is values that contribute to making and evaluating a decision. The actions are chosen based on the judgment of the values itself and not the rewards. Predicting these values is the most non-trivial process as this includes a series of observation and estimation over the agent's existence in the environment until the estimates are efficiently calculated. The fourth is the model of the environment which gives an overview of the surrounding itself.

There are two methods in how RL problems are tackled, one is model based and the other is model free based. The model based involves in solving RL related problems that use models and plans whereas model free based simply uses trial and error methods. Usually a model based is considered to be a planning method and model free as learning.

2.4.1 Model Based vs Model Free

The problems that revolve around RL systems include a model of the environment which is a comprehensive view of how the environment behaves for a respective state and action. The model of the environment is used for planning, which means for a respective state, a calculated decision is made on what actions to be taken considering the possible future situations before the actions are summoned. Models that use models and planning are called model based methods and on the contrary, there exists model free methods which uses trial and error learners.

In a model based problem, the agent should be able to predict the response from the environment for its actions. In other words, for a given state and actions, a model produces the prediction of the resultant next state and next reward, if the model is stochastic there will be a series of predictions for the resultant state and reward [5]. The model free method does not necessarily require a model of the environment, the model free systems cannot make calculative guess to predict the response of the environment for each action. In order to deal with a model based problem, the model

must be accurate hence acquiring an accurate model for a complex system is difficult which make model free methods more approachable.

2.5 BURLAP

BURLAP is a java code library, licensed under Apache 2.0 that is used for the development of single or multi agent planning and learning algorithms along with the domains to accompany them. It's a very versatile framework that caters to OMDP formalized problems such as defining states, actions and supporting discrete continuous and relational domains. It also supports planning and learning algorithms ranging from classic forward searching to value function based stochastic planning and learning algorithm [4]. A java documentation is provided for all the classes with a detailed description of their use in a domain.

2.6 BurlapCraft

Minecraft extends the use of mods which are modifications made to the environment based on the user's preferences. The mod, BurlapCraft [8] was used to deploy the algorithms present in Burlap. It includes scenarios such as dungeons that offers different levels of difficulty to test the learning and planning algorithms. In order to validate BurlapCraft as a platform for AI development the developers have tested Rmax, BFS, A*[8] algorithms. In concluding their research, the authors have stated that new techniques or new algorithms should be developed in order to cope up with the challenges that was put forward in the scenarios created in the BurlapCraft [8].

2.7 Learning & planning algorithms

The proposed algorithms that is to be tested for efficiency of the reinforcement learning and planning algorithms are Q learning, State Action Reward State Action(SARSA), IDA* and DFS developed in BURLAP.

2.7.1 Learning algorithms

2.7.1.1 Q Learning

It is an approach that allows the agent to learn how to act in an optimal way in a Markovian controlled domain. It is an incremental approach for dynamic programming and provides the agent to improve the quality of the particular action at the particular state by experiencing through consequences of actions without requiring the details of the domain [32].

Q Learning is derived from the model free approach which learns a policy or value function directly from experience. As mentioned before it learns the estimated Q values of an MDP in which the behavior can be dictated by taking the actions greedily with respect to the learned Q values.

The following pseudocode [33] summarizes the Q learning algorithm:

```
1. Initialize Q-values (Q(s, a)) arbitrarily for all state-action
   pairs.
2. For life or until learning is stopped...
3.     Choose an action (a) in the current world state (s) based
   on current Q-value estimates (Q(s,·)
4.     Take the action (aa) and observe the outcome state (s′) and
   reward (r)
5.     Update Q(s, a):=Q(s, a)+α[r+ γ max_a′ Q(s′, a′) − Q(s, a)]
```

The key steps in the above pseudocode are steps 3 and 5. There are many ways to choose the actions based on the estimates from step 3, but usually the policy that is used is ϵ-greedy policy. Generally, the policy that is used should have some randomness to it so that it promotes the exploration of the state space.

2.7.1.2 SARSA

State Action Reward State Action, (SARSA) is also a learning algorithm that is similar to Q learning. The difference between the two is that in SARSA is that the Q values are updated with respect to the immediate Q value that is acquired in the next action

unlike in Q learning which updates the Q values based on maximum Q-values obtained in the next state. The other difference is that difference is that at every time step, Sarsa will also update the Q-values for state-action pairs experienced previously in an episode with respect to the amount specified by λ and how long ago the experiences occurred. Define the below method to solve our task with Sarsa [33].

2.7.2 Planning algorithms

2.7.2.1 Iterative deepening A star (IDA star)

One of the most optimal search based planning algorithms is IDA* that uses a predefined model but incorporates a reward function. It also uses an admissible heuristic that is provided in BURLAP [33]. IDA* is a graph traversal algorithm and a path search algorithm in which the shortest path can be deduced between the designated start node and the goal nodes. There is a slight variation of the iterative deepening depth first search in which it uses the concept of applying heuristic function to evaluate the cost to get to the goal from the A star algorithm. Since it is a depth first search algorithm, the memory consumption is less than that of A star [34].

2.7.2.2 Depth First Search

Most common search-based planning algorithm is DFS, it uses the concept of backtracking which searches all the nodes by going ahead. In other words, the algorithm visits each nodes that is on its way and where there are no other nodes in its current path it back tracks and visits all the nodes until all the unvisited nodes have been traversed [34].

Summary

Performing human like behavior does require capabilities such as planning and learning. These abilities are transformed into algorithms in BURLAP library, ranging from classic forward searching to value function based stochastic planning and

learning algorithm. These algorithms are deployed in Minecraft, a game that provides the simulation and complexity that of a real world.

The mod, BurlapCraft [8] is used to deploy the algorithms present in Burlap. It includes scenarios such as dungeons that offer different levels of difficulty to test the learning and planning algorithms. In order to validate BurlapCraft as a platform for AI development the developers have tested Rmax, BFS, A star algorithms.

In concluding their research, the authors have stated that new techniques or new algorithms should be developed to cope up with the challenges that was put forward in the scenarios created in the BurlapCraft [8].
Despite these experiments there exist areas for development especially in minimizing the time the agent takes to learn and plan in Minecraft. The researchers [8] also recommend developing additional algorithms to solve the task oriented problems in Minecraft efficiently and effectively. But developing algorithms without exploring the existing ones in BURLAP with respect to virtual worlds such as Minecraft would be in vain and hence there should be further experiments conducted on other algorithms in BURLAP that may lead to a discovery of algorithms that are more effective and efficient in generating an optimal policy

CHAPTER 3

3 EMBEDDING ALGORITHMS IN BURLAPCRAFT

This research focuses on testing the algorithms in BURLAP on a scenario that is designed in Minecraft and evaluating the efficiency of the algorithms based on the time it takes to generate an optimal plan to complete the task.

BurlapCraft, the mod that is used as a test bed [8] for artificial intelligence is integrated with the BURLAP library. It is modelled in a way to perform various tasks within the game. BurlapCraft uses the Minecraft Forge API, which contains hooks into Minecraft to transport information between the Minecraft world and BURLAP. In order to validate BurlapCraft as an AI platform, the learning and planning algorithms are deployed and tested for their efficiency.

In this research, we test the efficiency of the reinforcement learning and planning algorithms. The algorithms that are tested are Q learning, State Action Reward State Action (SARSA), IDA* and DFS developed in BURLAP using the mod, BurlapCraft. We also analyze the effects of applying a model-free reinforcement learning algorithm such as Q learning in order to test its efficiency compared to that of model based planning algorithms.

This chapter provides an overview of the details of how the information is channeled and processed by the agent in Minecraft. The Section 3.1 describes the architecture and the internal details of the BurlapCraft mod along with the algorithms that was tested. Section 3.2 discusses how the agents are manipulated in BurlapCraft. Section 3.3 will discuss the algorithms used along with the implementation details.

3.1 Architecture of BurlapCraft

In order to understand the implementation details of the algorithms Burlap, it is important to understand the internal details of the BurlapCraft [7] which is shown in Figure 1.

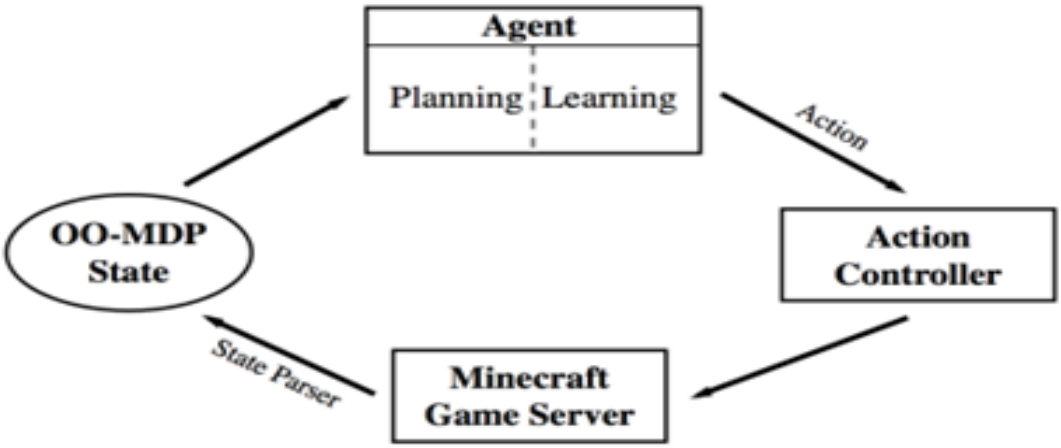

Figure 3.1: Overview of the BurlapCraft System Design [7]

The system shown in Figure 1 uses the standard game Minecraft server. The system observes the environment at any given time and converts it into an OOMDP state representation. This state is then passed to the agent that uses one of the planning/learning algorithms which enables the agent to choose an action. The action that is selected is then passed to the action controller for the executing the action in the environment. This process repeats until the expected reward function is reached. These rewards are user -defined as there is no intrinsic rewards or goals in Minecraft [7].

3.2 Manipulation of agents

The agents present in Minecraft are manipulated through the learning and planning algorithms. The environment's states are parsed into an OOMDP state and later passed to the agent. This gives a high-level information of the situation present in the environment to the agent. Based on this information is how the agent decides on what action is to be taken. The chosen action is passed to a low-level action controller which executes the respective action in the Minecraft server.

The agents are usually implemented by either learning or planning algorithms. If the agent uses learning algorithms, it will either learn their own model of the world and choose actions accordingly also known as model based reinforcement learning technique or directly learn the responses it gets when it chooses a particular action and based on the reward function it decides on which action to select, also known as model free reinforcement learning approach.

3.3 Planning and learning algorithms in Minecraft

The major part of this research focused on embedding the planning and learning algorithms, Q learning implemented in two approaches, SARSA, DFS and IDA star in BURLAP through BurlapCraft. These algorithms were tested for their efficiency in the predefined dungeons. Each of the algorithms was tested through the agents to solve the tasks given in Minecraft.

The reason for choosing Q learning and SARSA in the learning algorithm category is that both belong to the model free approach and does not require an accurate model which is suitable for the real world scenario. The difference between Q learning and SARSA is that Q learning is an off policy algorithm and SARSA is an on policy algorithm. The IDA star and DFS belong to the planning algorithms in which these two are preferred as these are low on memory consumption and unlike A star, IDA star only visits the potential nodes and not the all the surrounding nodes. The same applies to DFS as well. The only difference is that IDA star uses a heuristic function.

This chapter presents the implementation of the algorithms that were used on the agents to increase the efficiency in solving the task. It discusses about the initializing BurlapCraft in order to use it as a test bed for the algorithms and the following Section discusses about the initializing of BurlapCraft and deployment of the algorithms in BurlapCraft. Each of the algorithms along with the implementation is shown below.

3.3.1 Initializing BurlapCraft

The mod is run through Minecraft in the "creative" mode where there exist dungeons of different levels of difficulty. Invoking features of BurlapCraft is done through Minecraft's chat/command system [33].

3.3.2 Deploying Algorithms in BurlapCraft

The algorithms developed in the BURLAP library is tested in the BurlapCraft mod in which the algorithms, Astar, Rmax and BFS have already been tested. This research focuses on testing algorithms, IDA star, Q Learning, SARSA and DFS in BurlapCraft.

The algorithms, IDA star and DFS are categorized as the planning algorithms and Q Learning and SARSA as learning.

3.3.2.1 Testing of Q learning

Testing of Q learning algorithm uses two approaches. Approach 1 includes QLearning that is an instance of the LearningAgent interface which uses a predefined set of parameters unlike Approach 2, in which the Q learning algorithm is implemented with respect to updating the Q values and implementing the learning algorithm itself without only using the predefined LearningAgent interface.

Approach 1

The pseudocode for the algorithm for testing Q learning algorithm in Minecraft is as follows:

```
agent = new QLearning(lastDomain, 0.99, new
      SimpleHashableStateFactory(), 0., 1.);
List<Episode> episodes = new ArrayList<Episode>(1000);
for (int i = 0; i < 1000; i++){
episodes.add (agent.runLearningEpisode (me));
}
```

Algorithm1: Testing Q learning Algorithm in Approach 1

In Algorithm 1, the LearningAgent instance provides methods for learning with the environment. QLearning is an instance of the LearningAgent interface and takes parameters for domain, discount factor, SimpleHashtableStateFactory, an initial value

for the Q values and a learning rate. The constructor uses a default policy of 0.1 epsilon greedy policy.

To run a learning episode, the method runLearningEpisode on the LearningAgent instance is called and passes it to the Minecraft Environment in which the learning will be performed. In order to examine the record of interactions the Episode object is returned which is similar to the policies.

Approach 2

In this approach the algorithm actually implements the Q learning unlike the previous approach. It is crucial in getting and storing Q-values in the Q learning algorithm. The primary date that needs to be stored is the estimated Q value for each state and action pair. Once the Q value function is initialized and the learning rate parameter is set a learning policy will be followed in which the policy dictates how the agent chooses actions at each step.

The pseudocode for updating the Q values is as follows.

```
check if Q value is already stored
if(Q value == null){
      for(Actions a: actions){
      create a Q-value for each action
      add q with the initialized action
}store the Q values for the next state
```

Algorithm 2: Updating of Q values in Q Learning in Approach 2

The two approaches show the different ways to implement Q learning. In Algorithm 2, the Q learning algorithm is run through the agents in Minecraft for each of the 1000 episodes of learning. For each learning progress the agent should be able to get better at solving the task. Also, the agent's actions will be random as it follows an epsilon greedy policy.

3.3.2.2 Testing of SARSA

As stated in the previous section 3.3.2.1, SARSA is similar to Q learning but in

SARSA the Q values are updated based on the next action taken rather than the maximum value. The SarsaLam instance is constructed and the parameters of learning rate is 0.5, and the λ value is 0.3.

The pseudocode for the algorithm for testing SARSA algorithm in Minecraft is as follows:

```
agent = new SarsaLam(lastDomain, 0.99, new
       SimpleHashableStateFactory(), 0.,0.5,0.3);
      for(int i = 0; i < 1000; i++){
                Episode e = agent.runLearningEpisode(me);
}
```

Algorithm 3: SARSA Algorithm

3.3.2.3 Testing of IDA star

The IDA star algorithm uses admissible heuristic that estimates the cost to the goal from any state similar to the A star algorithm. The heuristic is implemented using the Manhattan distance to goal heuristic. The deterministic planner is instantiated to IDA star.

The pseudocode for testing IDA star algorithm in Minecraft is as follows:

```
Heuristic mdistHeuristic = new Heuristic() {

double mdist = Math.abs(a.x-gx) + Math.abs(a.y-gy) +
       Math.abs(a.z-gz);
return -mdist;
                     }
planner = new IDAStar(domain, gc, new
       SimpleHashableStateFactory(),mdistHeuristic);
```

Algorithm 4: IDA star Algorithm

3.3.2.4 Testing of DFS

An instance of the DFS planning algorithm is created which is a subclass of DeterminsiticPlanner class.

The pseudocode for testing DFS algorithm in Minecraft is as follows:

```
DeterministicPlanner planner = new DFS(domain, goalcondition,
                                new SimpleHashableStateFactory());
```

Algorithm 5: DFS Algorithm

In order to instantiate DFS, it requires a reference to the domain, the goal condition for which it should search and the `SimpleHashableStateFactory` object that specifies how to hash and check state equality for states.

3.4 Summary

The learning and planning algorithms, Q learning, SARSA, IDA star and DFS present in the BURLAP library are tested using BurlapCraft. An overview of the BURLAP architecture is discussed. The algorithms are applied to the agents and then later deployed in Minecraft. An introduction of the algorithms to be tested are also given. The learning algorithm is similar to that of the planning algorithms except that in learning algorithms, the agent has to interact with the environment which causing multiple episodes of learning to be run. The Q learning algorithm is tested using two approaches. The SARSA algorithm is similar to Q learning but in SARSA the Q values are updated based on the next action taken rather than the maximum value. The IDA star uses admissible heuristic to estimate the cost to the goal from any state. DFS algorithm is created through an instance of the DFS planning algorithm which is a subclass of DeterminsiticPlanner class.

CHAPTER 4

4 EVALUATION

To demonstrate the effectiveness of the work, an experiment was performed in Minecraft through the agent that utilized the planning and learning algorithms.

The main goal of this research is to find the possibility of extracting efficient algorithms that is better than the algorithms that are presented in the literature. This research uses two aspects to quantify the efficiency of the algorithm. The first one is the completion of task by the agent. The agent deployed in the respective dungeon must complete the task in order to be considered as efficient. The other factor is the time the agent takes to complete the task. In this research the learning and planning algorithms are applied through the agents to complete the task assigned in the specified dungeon. The dungeons are of different caliber that are later mapped to problems present in the real world. The dungeons include, grid dungeon, maze dungeons, bridge dungeon and finder dungeon.

The performance evaluation is based on the following. The first is the comparison of the Rmax, A star and BFS algorithms that were tested in the existing experiments using BurlapCraft. The second analyses the impact the algorithms made on the performance of the agent in completing the assigned task. Based on the evaluation, the algorithms that are best fit and worst fit for the respective dungeons will be identified.

This chapter presents the comparison of both the existing algorithms and the work conducted in this research. The planning algorithms, IDA star and DFS will be tested along with the previously tested planning algorithms, A star and BFS. The learning algorithms, Q learning and SARSA will be tested along with the previously tested learning algorithm, Rmax.

4.1 Experiment Setup and Methodology

In this section multiple learning and planning algorithms are explored within the Minecraft through BurlapCraft. Through the agent the planning and learning algorithms are deployed in the predesigned dungeons.

The planning algorithms, IDA star and DFS are applied to the agents in Minecraft to solve the tasks associated in each of the dungeons. There exist three dungeons, each of which is on different levels of difficulty in BurlapCraft [8].

Bridge dungeon: This is an area that encloses dimensions of 10x10x5 including a mineable block, a gold block that is separated by a lava. The task that the agent has to solve is to reach the goal that is on the other side of the lava and in order to do so the block needs to be mined and placed on top of the lava [8]. Testing the agent in this dungeon compensates for solving tasks in a complex situation.

Easy maze dungeon: This is an area that encloses an area of 14x14x4 with a task that includes an agent to reach the goal which is a gold block from the furthest point. The finder dungeon is similar to that of the easy maze but comprises of a smaller area that is suitable for testing learning algorithms, Q learning, SARSA and Rmax that consumes more time.

Grid dungeon: This is an area that encloses an area of 5x5x3 grid modeled dungeon that includes a task of reaching the gold block from the initial point.

The efficiency of the planning and learning algorithms are measured through deploying the agent in Minecraft and extracting the time the agent takes to complete the task. The agent depending on the algorithm it possesses would complete the respective task in period of time for the given dungeon.

For example, if the IDA star algorithm is applied to the agent present in the maze dungeon, then the agent will be expected to complete the task of reaching the gold block from the furthest point. The time taken for the agent to complete this respective task is recorded. This process is repeated for ten trials for accuracy and then the average is calculated. The average time is evaluated for the efficiency of each algorithm. This procedure is repeated for planning algorithms, IDA star, A star, BFS, DFS and learning algorithms, Q learning, SARSA and Rmax.

Through the results the average time(s) is deduced and the efficient algorithm will be designated based on the minimum average time consumed to complete the task by the agent possessing the respective algorithm.

4.2 Performance

Planning Algorithms: Performance of the agent in the respective dungeon

4.2.1 Agent deployed in Bridge Dungeon

Table 4.1 represents the results of the agent that is deployed in the Bridge dungeon. The agent should complete the task of reaching the goal that is on the other side of the lava and to do so the agent should mine the block and place the mined block on the surface of the lava. Since the mined block acts as a bridge, the agent will now be able to go to the other side of the dungeon. Once the agent is on the other side of the bridge the task is assumed to be complete.

4.2.1.1 Results

Table 4.1: Time taken to complete the task in Bridge Dungeon

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Average(s)
IDA Star	16.534	16.541	16.548	16.538	16.539	16.535	16.536	16.541	16.537	16.540	16.539
A Star	16.539	16.540	16.556	18.059	18.097	16.554	18.056	16.560	16.537	16.542	17.004
BFS	16.540	16.544	16.569	16.547	16.543	16.543	16.551	16.540	16.567	16.548	16.549
DFS	16.541	16.545	16.541	16.541	16.544	16.534	16.569	16.545	16.563	16.543	16.547

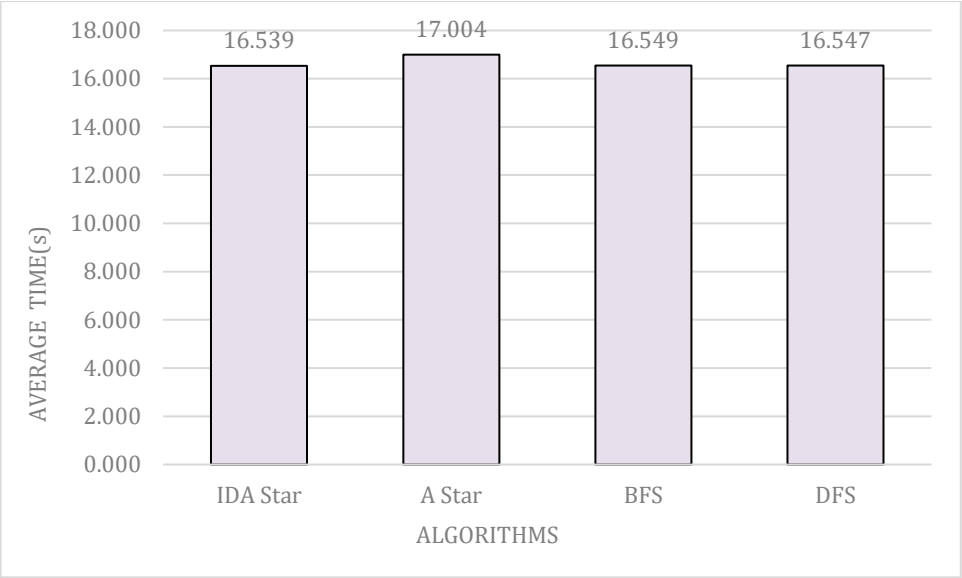

Figure 4.1: Time taken to complete the task in Bridge Dungeon

According to Figure 4.1 the algorithms can be ranked as follows: IDA star has the minimum average time of 16.539 and hence it is ranked as the first, the second to follow would be DFS, average time of 16.547, third is BFS, average time of 16.549 and the last would be A star having a time of 17.004.

4.2.2 Agent deployed in Easy Maze Dungeon

Table 4.2 represents the results of the agent that is deployed in the Easy Maze Dungeon. The agent should complete the task of reaching the gold block placed in the modelled maze. The agent should traverse the network of paths and hedges of the maze in order to reach the gold block. The task will be assumed as complete once the agent reaches the gold block.

4.2.2.1 Results

Table 4.2: Time taken to complete the task in Easy Maze Dungeon

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Average(s)
IDA Star	58.593	58.596	58.601	58.625	58.614	58.651	58.668	57.105	58.647	58.686	58.479
A Star	58.685	58.605	58.586	58.591	58.592	58.582	58.593	58.592	58.591	58.586	58.600
BFS	58.595	58.630	58.620	58.641	58.982	58.593	58.614	58.604	58.635	58.614	58.653
DFS	58.585	58.597	58.595	58.610	58.628	58.628	58.628	58.589	58.610	58.622	58.609

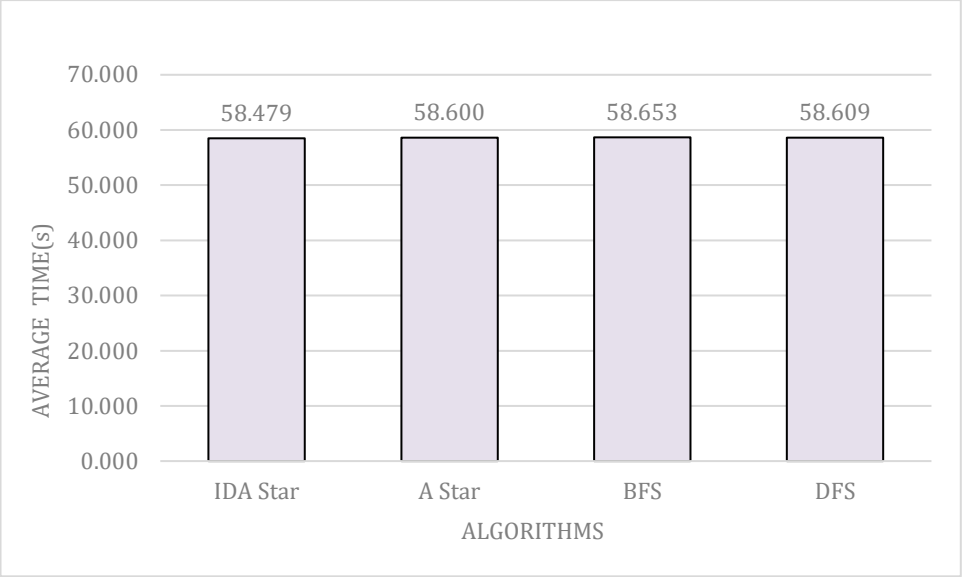

Figure 4.2: Time taken to complete the task in Easy Maze Dungeon

According to Figure 4.2 the algorithms can be ranked as follows: IDA star has the minimum average time of 58.479 and hence it is ranked as the first, the second to follow would be Astar, average time of 58.600, third is DFS, average time of 58.609 and the last would be BFS having an average time of 58.653.

4.2.3 Agent deployed in Grid Dungeon

Table 4.3 represents the results of the agent that is deployed in the Grid Dungeon. The agent should complete the task of reaching the gold block placed in the grid. The agent should traverse along the wall to reach the gold block. The task will be assumed as complete once the agent reaches the gold block.

4.2.3.1 Results

Table 4.3: Time taken to complete the task in Grid Dungeon

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Average(s)
IDA Star	22.569	22.539	22.544	24.059	22.545	22.542	22.572	22.543	22.539	22.540	22.699
A Star	22.552	22.540	22.570	22.539	22.548	25.591	22.453	22.553	25.591	22.586	23.152
BFS	22.540	24.039	22.540	18.303	22.543	22.602	22.586	22.539	22.566	22.540	22.280
DFS	22.553	22.562	22.539	22.540	22.542	22.555	22.551	22.545	22.541	22.543	22.547

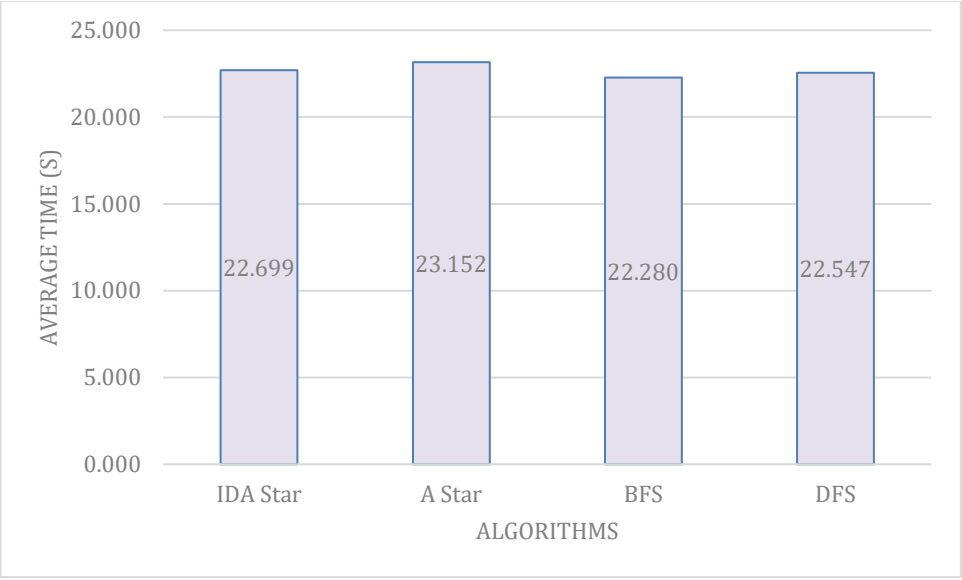

Figure 4.3: Time taken to complete the task in Grid Dungeon

According to Figure 4.3 the algorithms can be ranked as follows: BFS has the minimum average time of 22.280 and hence it is ranked as the first, the second to follow would be DFS, average time of 22.547, third is IDA star, average time of 22.699 and the last would be A star having an average time of 23.152.

4.2.4 Agent deployed in Finder Dungeon

Table 4.4 represents the results of the agent that is deployed in the Finder Dungeon. The Finder dungeon is used to test the learning algorithms such Q learning: Approach 1 and Approach 2. Rmax and SARSA. The finder dungeon is similar to that of the easy maze but comprises of a smaller area that is suitable for testing learning algorithms as unlike planning algorithms, these consumes more time. The task is also similar to that of the easy maze dungeon in which the agent has to traverse the network of paths and hedges of the maze in order to reach the gold block. The task will be assumed as complete once the agent reaches the gold block.

4.2.4.1 Results

Table 4.4: Time taken to complete the task in Finder Dungeon

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Average (s)
Q Learning: Approach 1	161.304	56.883	131.522	21.709	158.348	55.091	49.328	126.928	158.351	133.419	105.288
Q Learning: Approach 2	135.339	215.367	277.488	55.219	77.788	121.49	169.625	120.135	120.135	81.123	137.371
SARSA	137.409	105.663	38.128	118.944	106.317	70.38	160.037	116.985	164.83	153.712	117.241
Rmax	54.236	61.757	156.71	25.624	43.696	87.492	69.251	57.222	99.422	94.967	75.038

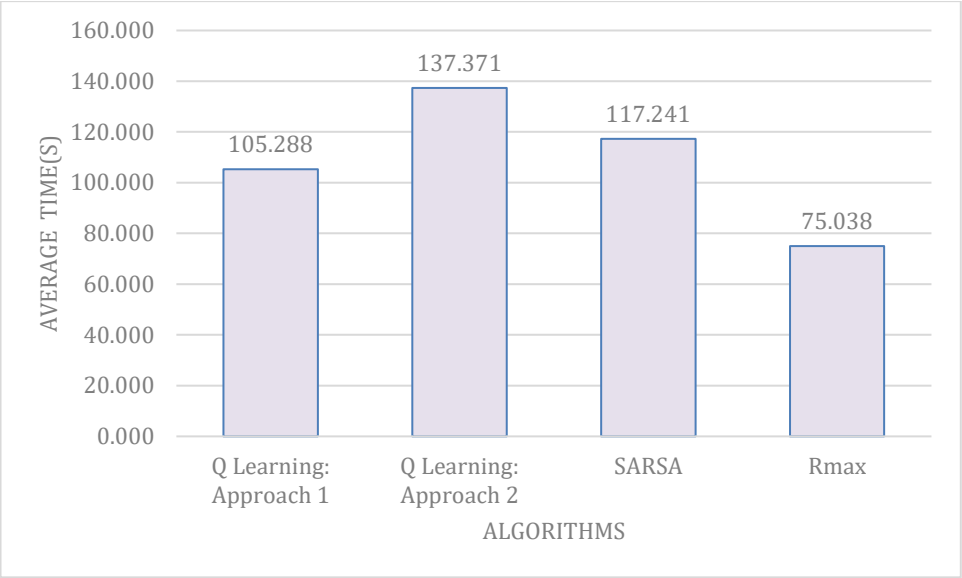

Figure 4.4: Time taken to complete the task in Finder Dungeon

According to the results shown in Figure 4.4 the algorithms can be ranked as follows: Rmax has the minimum average time of 75.038 and hence it is ranked as the first, the second to follow would be Q learning: Approach 1, average time of 105.288, third is SARSA, average time of 117.241 and the last would be Q learning: Approach 2 having an average time of 137.371.

4.3 Discrepancies

While the algorithms were deployed in Minecraft through the agent there were few complications. When the IDA star algorithm was deployed in the Bridge Dungeon, in-between trial 4 and 5, the agent got stuck when it got to the block that was to be mined. Therefore, the program was resumed, and the other results of the trials were recorded with no issue. This issue occurred only once. In the case of BFS, A star and DFS the inconsistencies were serious compared to that of the IDA star. The agent possessing algorithms such as A star was not able to calculate the distance it should place the block on. The block was thrown to the other end of the lava which is a serious issue compared to that of the agent possessing IDA star.

There weren't any issues regarding the other Dungeons, the agent completed the task of the respective dungeons without any discrepancies.

Though the algorithms did have a few inconsistencies, overall the agent did complete the assigned task for the respective dungeon. Therefore, knowing the strength and weakness for the scenario is important to produce efficient results through the algorithms.

4.4 Discussion

A dungeon in Minecraft represents a problem in real life. Each dungeon includes tasks that are of different caliber. The task assigned to Bridge dungeon could be compared to a complex decision-making problem as the agent should make a calculative decision in mining the block and placing it on the lava itself. This scenario can be applied to different situations in the real world, it need not be the exact scenario of mining and placing it on the lava, it could be any scenarios that involves complex decision making. The other dungeon, easy maze includes the task of reaching the gold block placed in the modelled maze. This again could be applied to a search problem in the real world, for example bomb diffusing scenario, the agent should traverse the area to locate the bomb that is to be detonated. Similar concept can be applied to the finder dungeon that includes the same task enclosed in a smaller area. The final dungeon is the grid dungeon which could be applied to a general scenario or rather any scenario. There are many problems that include traversal for example a mundane task of travelling from point A to point B could be represented by the dungeon hence the grid dungeon represents both major and minor issues.

The results shown in Table 4.1, 4.2, 4.3, and 4.4 indicate the performance of the algorithms and it could be further analyzed to indicate which of the algorithms work best for a scenario and these algorithms could be further applied to resolve the real-life problem of the respective category.

Since solving a complex task is represented by the Bridge Dungeon. It is possible to analyze the algorithms and select the best fit algorithms for solving complex tasks. The Figure 4.1 shows the most efficient algorithm in solving complex tasks is IDA star which also has been consistent compared to the other algorithms, BFS, DFS and A star. Though the agent got stuck in the middle of the program, it may be due to the system and not due to the algorithm itself. Also throughout the trials the agent made the correct estimate to place the block on the lava completing the assigned task successfully. The algorithms DFS could also be a satisfactory since the difference in time is only 0.10s. Also, there were inconsistencies in DFS which makes IDA star the best algorithm and the worst algorithm to handle complex task would be A star as it has consumed the maximum amount of time.

The Easy maze bridge dungeon represented the scenario of a search problem. It is possible to do the same procedure of analyzing the algorithms that was carried to these as well. Again, Figure 4.2 shows the most efficient algorithms in solving a search problem is IDA star compared to the other algorithms tested. Problems that include both these scenarios could be resolved through the IDA star algorithm. Figure 4.2 also indicates the worst algorithm for using for a search problem is BFS.

The Grid dungeon represents a general type of scenario in which the agent should traverse the dungeon which is an open area and find the goal. This is applicant to any traversal problem, and Figure 4.3 shows the efficient algorithm to do perform such task includes BFS and the worst algorithm is A star. The results shown in Table 4.3 indicate that the efficient is indeed BFS when compared to other algorithms as the extracted time is relatively low.

The Finder dungeon is a minor version of the Easy Maze dungeon, in which in this dungeon the learning algorithms are tested. The learning algorithms include Rmax, the two approaches in the two Q learning algorithms and then the SARSA algorithms. Figure 4.4 depicts the most efficient algorithm to be is Rmax and the worst is Q learning algorithm implemented in approach 2. The results do indicate that Rmax is best fit to solve such scenarios, but it is important to note that Rmax is categorized under model based learning algorithm [35] hence it explores the environment, then

learn the model and use the model to plan the policy. This approach is efficient and hence the results shown in Table 4 indicate as such. But the second most efficient algorithm is Q learning implemented in approach 2, this algorithm is categorized under model free algorithm[35] hence it does not learn the model, it learns the value function or policy directly and thus leading to weaker results compared to that of Rmax. But the model based learning is effective only when the state space is manageable in other words it may not be well suited for complex task in the real world. However, the model free approach, is efficient when it comes to a larger state space. Therefore, the results depicted in Table 4 is extracted from the Finder dungeon that is as mentioned before is a smaller area. In other words, the finder dungeon comprises of a manageable state space. Since the second-best algorithm indicated in the Figure 4.4 is Q learning (approach 1) algorithm it is safe to state that this algorithm may indeed work better than that of the Rmax algorithms.

Complexity analysis of algorithms is the cost, measured in running time or storage or the units that are relevant of using the algorithm to solve a problem [36]. Since this research focuses on evaluating the performance it is important to discuss the complexities for each of the algorithm that were tested.

DFS is a well renowned recursive algorithm that uses the idea of backtracking for a problem that involves traversing or searching. When we apply this concept to the states present in Minecraft, the states are traversed exactly once and therefore the complexity of it is O(n), where n is the number of states. If an action that leads to the state is already traversed then that particular state is skipped and the next state is traversed. IDA star provides a low space complexity but completeness and optimality is the key idea of it. It is implemented using DFS to look for the goal at each layer of states. In other words it is a depth bounded DFS search. Also in this scenario it uses a heuristic based IDA star in which if you cannot find the goal the depth bound will be increased. The time complexity of IDA star is O (b^n), where b is the branching factor [34].

The complexity for learning algorithms differs with that of the planning algorithms as complexity deducing in learning algorithm is based off a model free technique in

which the model is unknown in order to deduce the complexities. Q learning and SARSA are algorithms that follow a procedure in which an agent finds its way to one of the set of goal locations through actions that would take the agent from one state to another. The algorithm is such that in the initial stage the agent is not aware of the topology of the state space and it only discovers about it through exploration. In literature it is stated that reaching a goal state through reinforcement learning techniques may require number of action executions and therefore the state space is exponential in size [37]. In order to deduce the worst case complexity of reaching the goal of an uninformed algorithm such as Q learning and SARSA the agent must learn something about the consequences of the action it takes in order to have a complexity that is less than infinity. For example assume Q learning is initialized to zero and operates on a goal-reward representation. For each action the agent takes, the Q values that lead to the goal state only changes. The other Q values remain zero and since the action selection step does not provide any information on the undirected exploration the agent has to choose actions based on the random walk and hence the average number of steps required for the agent to reach the goal is exponential in *n*, the number of states [38]. This complexity can be applied to the SARSA algorithm as well as it also follows the similar execution of the Q values with respect to updating it based on the executed action values only and hence the complexity could be exponential in *n*, the number of states.

CHAPTER 5

5 CONCLUSION

This research is focused on applying the reinforcement learning and planning algorithms developed in BURLAP on a pre-designed scenario that is modelled using OOMDP in Minecraft. It investigated the potential of learning and planning algorithms present in BURLAP which lead to the discovery of algorithms that are more effective and efficient in generating an optimal policy. The proposed algorithms that were tested for efficiency are Q learning, SARSA, IDA* and DFS.

The main goal of this research is to find the possibility of identifying efficient algorithms that is better than the algorithms that is present in the literature. This research uses two aspects to quantify the efficiency of the algorithm. The first one is the completion of task by the agent. The other factor is the time the agent takes to complete the task. In this research the learning and planning algorithms are applied through the agents to complete the task assigned to the respective dungeon. The dungeons are of different caliber that are later mapped to problems present in the real world. The dungeons include, grid dungeon, maze dungeons, bridge dungeon and finder dungeon.

In conclusion this work proves that the algorithms, IDA star and Q learning (approach 1) that were not tested in the existing literature do make an impact in improving the efficiency of the agent in completing specified task. The dungeons where the algorithms were tested are mapped to similar problems present in the real word which gives an understanding of the potential of the algorithm. This work also identifies the best fit and the worst fit algorithms for the respective dungeons. Each of these dungeons could be mapped to general problems such as decision making, search and traversal (from point A to point B) and based on these the best fit algorithm that worked for these scenarios could be chosen, likewise the worst fit could be avoided to resolve similar problems existing in the real world.

5.1 Future Work:

We plan to extend our study on following directions:

5.1.1 Evaluating the potential of Q learning algorithm in a decision-making problem

This work proposed the testing of the Q learning algorithms in the Finder dungeon which comprised a menial task unlike the task assigned to the bridge dungeon. In future the dungeon should be assigned a task similar to that of the Bridge dungeon so that Q learning algorithm could be tested for its efficiency in a decision making problem. Therefore the performance evaluation would state if the agent is efficient to make a decision in resolving the problem.

5.1.2 Evaluating the potential of Q learning algorithm in a larger state space

This work proposed the testing of the Q learning algorithms in the Finder dungeon which is a smaller area and hence the potential of its performance in a larger state space remains unknown. It is stated that the Q learning algorithms work better in a larger state space than that of the model based learning algorithm, Rmax. Therefore a scenario comprising of a larger state space must be designed in order to test both Q learning and Rmax. The performance evaluation would confirm that Q learning in fact would be better at solving tasks that are similar to that of the real world.

REFERENCES

1. J. M. Bradshaw, Software Agents. Cambridge: AAAI Press/MIT Press, pp. 7-30 1997.
2. M. Wooldridge and N. R. Jennings, “Intelligent agents: theory and practice”, The Knowledge Engineering Review, vol. 10, no. 02, p. 115, 1995.
3. S. J. Russell and P. Norvig, Artificial Intelligence. "A modern approach." Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25, pp. 32-58, 1995.
4. “About,” BURLAP. [Online]. Available: http://burlap.cs.brown.edu/. [Last Accessed: 2017-12-27]
5. R. S. Sutton and A. G. Barto, “Reinforcement learning: an introduction”. Cambridge, MA: The MIT Press, 2012
6. Official site,” Minecraft.net. [Online]. Available: https://minecraft.net/. [Accessed: 06-Apr-2017].
7. C. Diuk, A. Cohen and M. Littman, "An object-oriented representation for efficient reinforcement learning", Proceedings of the 25th international conference on Machine learning - ICML '08, pp 240-247, 2008.
8. K Aluru, S Tellex, J Oberlin, J MacGlashan, "Minecraft as an experimental world for AI in robotics." In AAAI Fall Symposium. 2015.
9. B.E. Mennecke, D. McNeill, M. Ganis, E.M. Roche, D.A. Bray, B. Konsynski, A.M. Townsend and J. Lester, "Second Life and Other Virtual Worlds: A Roadmap for Research", Communications of the Association for Information Systems, 22, 20, pp. 371-388, 2008.
10. N. Yee, “The Demographics, Motivations, and Derived Experiences of Users of Massively Multi-User Online Graphical Environments,” Presence: Teleoperators and Virtual Environments, vol. 15, no. 3, pp. 309–329, 2006.
11. P. A. Fishwick, “An introduction to OpenSimulator and virtual environment agent-based M&S applications,” Proceedings of the 2009 Winter Simulation Conference (WSC), pp 177-183, 2009.
12. J. Moline, “Virtual environments for health care”. A white paper for the Advanced Technology Program (ATP) National Institute of Standards and Technology. R, 1995.
13. M. Luck and R. Aylett, “Applying artificial intelligence to virtual reality: Intelligent virtual environments,” Applied Artificial Intelligence, vol. 14, no. 1, pp. 3–32, 2000.
14. K.A. Thomsen, D.C. Rasmus and D. Jensen, “Smart Dog for Minecraft”, 2014
15. A. Bahrammirzaee, “Contribution to study and design of intelligent virtual agents: application to negotiation strategies and sequences simulation.” Diss. Université Paris-Est, 2010.
16. F. Brazier, C. Jonker, and J. Treur, “Compositional design and reuse of a generic agent model,” Applied Artificial Intelligence, vol. 14, no. 5, pp. 491–538, 2000.
17. B. Detlor, “Intelligent Agents and Knowledge Portals,” Towards Knowledge Portals Information Science and Knowledge Management, pp. 147–173, 2004.
18. M. J. Wooldridge and N. Jennings, “Intelligent agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages”, Amsterdam, the Netherlands, August 8-9, 1994: proceedings. Berlin: Springer-Verlag, 1995.

19. N. R. Jennings, M. J. Wooldridge. “Applications of intelligent agents”. In: N. R. Jennings, M. J. Wooldridge (eds.), Agent Technology: Foundations, Applications, and Markets, Springer, pp. 3–28, 1998.
20. P. Patel, H .Hexamoor. “Designing BOTs with BDI agents, In: International Symposium on Collaborative Technologies and Systems (CTS) Carbondale”, USA, pp. 180–186, 2009.
21. J. M. Vidal. “Fundamentals of multiagent systems with netlogo examples”, 2006. [Online]. Available: http://www.multiagent.com/fmas. [Accessed: 06-Apr-2017].
22. A. Bogdanovych, J. A. Rodriguez-Aguilar, S. Simoff, and A. Cohen, “Authentic Interactive Reenactment Of Cultural Heritage With 3D Virtual Worlds And Artificial Intelligence,” Applied Artificial Intelligence, vol. 24, no. 6, pp. 617–647, Dec. 2010.
23. J. Gemrot, C. Brom, and T. Plch, “A Periphery of Pogamut: From Bots to Agents and Back Again,” Lecture Notes in Computer Science Agents for Games and Simulations II, pp. 19–37, 2011.
24. B. Damar, "Avatars! Exploring and building virtualworlds on the Internet". Berkeley, CA: Peachpit Press, 1998.
25. J. C. B. P. Müller, “The design of intelligent agents: a layered approach”. Vol. 1177. Springer Science & Business Media, 1996.
26. F. Bálint-Benczédi, T. Wiedemeyer, M. Tenorth, D. Beßler., and M. Beetz “A Knowledge-Based Approach to Robotic Perception using Unstructured Information Management." Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, pp.1941-1942, 2015.
27. R. Hanna "Kant's Theory of Judgment", The Stanford Encyclopedia of Philosophy,Edward N. Zalta (ed.), 2016, forthcoming URL = <https://plato.stanford.edu/archives/win2016/entries/kant-judgment/>. [Online; last accessed 12-03-2017].
28. D. J. Chalmers, R. M. French, and D. R. Hofstadter, “High-level perception, representation, and analogy: A critique of artificial intelligence methodology,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 4, no. 3, pp. 185–211, 1992.
29. C.T. Morrison, E. Dietrich. "Structure-mapping vs. high-level perception: The mistaken fight over the explanation of analogy." Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, pp. 678-682, 1995.
30. O. Sýkora, “State-space Dimensionality Reduction in Markov Decision Processes.”, 2008.
31. S. Thrun, W. Burgard, and D. Fox, “Probabilistic robotics”. Cambridge, Mass.: MIT Press, 2010.
32. C. J. Watkins, P. Dayan, “Q-learning. Machine learning”, 8(3-4), pp. 279-292, 1992
33. J. MacGlashan, Brown-umbc reinforcement learning and planning (burlap), 2009. http://burlap.cs.brown.edu/. Online; last accessed: 2017-12-27.
34. R. E. Korf, “Depth-first iterative-deepening: An optimal admissible tree search. Artificial intelligence”, 27(1), pp. 97-109, 1985.

35. L.P. Kaelbling, M.L. Littman, and A.W. Moore, “Reinforcement learning: A survey”. Journal of artificial intelligence research, 4, pp.237-285, 1996.
36. S. H. Wilf, Algorithms and Complexity, University of Pennsylvania , Internet Edition, Summer, 1994
37. S.D. Whitehead, “A complexity analysis of cooperative mechanisms in reinforcement learning”. In Proceedings of the AAAI. 607–613, 1991.
38. S. Koenig, and R. G. Simmons. “Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains”. No. cmu-cs-93-106. Carnegie-Mellon University Pittsburgh PA School of Computer Science, 1992.