THINKING LIKE HUMAN APPROACH TO AI FOR GAMEPLAYING

Sadika Hasantha Sumanapala

(189398U)

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

March 2021

THINKING LIKE HUMAN APPROACH TO AI FOR GAMEPLAYING

Sadika Hasantha Sumanapala

(189398U)

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

March 2021

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and 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 to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/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).

Name of the Student:
S. H. Sumanapala

Signature of the Student: Date:

The above candidate has carried out research for the Masters thesis under my supervision.

Name of the supervisor:
Prof. A. S. Karunananda

Signature of the Supervisor: Date:

Dedication

I dedicate this thesis to my family and friends. I will always appreciate the help of friends for the things all they have done and their valuable thoughts. I dedicate this work and give many thanks to people at University of Moratuwa for their help and especially for lecturers for providing guidance throughout this research.

Acknowledgement

I would like to express my very great appreciation to Prof. Asoka Karunananda for his generous support for this project.

Also, I would like to thank all the lectures of the Department of Computational Mathematics for their guidance during the course and all the resources and support provided to make this project successful.

Abstract

Playing games helps humans relax and their minds. Games have a very long history. They were used as a leisure activity even by early humans. It involves both mind and body. With the development of computers games became more complex and entertaining. Computers are extensively used to model, simulate and develop games. Games require us to use our cognitive power to plan and take sudden discussions to win. Chess, Go are highly complex games in terms of options that a player must think about when making a move. There are a lot of different paths that a player can take to win a game. Therefore, many researchers try to use various technologies to develop applications which can play games. The development of these technologies also helps to solve complex problems in completely different areas such as finance, economics, and warfare.

According to the other researchers work. Multi Agent Systems (MAS) have ability to modal complex problems. Because of that we developed a system using MAS to play games. Suggested solutions are developed using thinking of human's behaviors. When humans face a problem, they think about it in several ways. In their mind they compare, contrast and argue to find the best possible solution. Similarly, in multi agent systems intelligent agents solve complex problems by communication, coordination and negotiation. Game playing is a problem solving that involves thinking, decision making and negotiation skills of the human mind. Even though this is very intuitive to the human mind proper designing needs to be done to model this way of problem solving into an AI system. Here we designed a multi agent game playing system to play N-puzzle game. It contains mainly two types of agents: coordinator agent and decision agents. For the N puzzle game there are four decision agents namely up, down, left, right decision agents. They analyze the game state, negotiate with each other to determine the best move to make. For the development of the solution, we used SPADE architecture-based multi agent framework. The results show the proposed solution able to achieve notable improvement (~42%) compared to human players.

Keywords: Agents, Multi Agent Systems, Game playing

Table of Contents

List of Figures

List of Tables

List of Abbreviations

MAS	Multi Agent System
API	Application Programing Interface

CHAPTER 1

INTRODUCTION

1.1 Prolegomena

Games are an integral part of human life from ancient times [1]. Every human culture has some kind of game embedded into their core [2]. There are many types of games, some involve the mind, some involve the body, and some games involve both mind and body.

Board games such as chess, checkers, Go highly utilize our mind. These kinds of games help us to relax our mind [3]. With the introduction of computers games also evolved. They became more complex and entertaining. Now computers are extensively used to model, simulate and develop games. Almost all games require us to use our cognitive power to plan and take sudden discussions to win or achieve the goal of the game.

Most games contain multiple paths to achieve the winning goal. For example, in puzzle games such as N-Puzzle or board games such as Chess, Go are highly complex in terms of options or moves that the player must select in each instance. Players must consider not only the current move but also a few future moves in order to make a successful decision. Many researchers are fascinated by the complexity offered by these game problems. They try to use various technologies to develop applications which can play games. The development of these technologies also helps to solve complex problems in completely different areas such as finance, economics, and warfare.

1.2 About Game Playing and Thinking

Thinking like a human is considered as one of the most influential schools of thought in Artificial Intelligence. Undisputedly, people have demonstrated their intelligence when they have been involved in tasks requiring thinking. Among other activities, gaming, problem solving, theorem proving, planning are well known examples where thinking is involved. Since the inception of AI, solutions coming under

thinking like humans have been implemented by symbolic AI approach to Artificial Intelligence. In line with that tradition, we postulate that Multi Agent technology, probably the champion of symbolic-AI, can be used to model game playing solutions. Inspiration for this hypothesis comes from the fact that the thinking process required for gaming in a mind mimics deliberation among different agents following certain rules. At a given state, a deliberation among the agents determines the best option/action for a player. As such, entities and/or locations in a game space can be modeled as agents by coding the rules for their behavior, interactions, and actions. This approach works better than the traditional state space approach and machine learning approach to AI. More importantly, our approach works beyond human capacity to game playing, where humans can visualize only a limited option available at a given stage. Since agents can deliberate in parallel, a multi agent approach works more efficiently than state space search.

In this project we have used multi agent technology to develop game player systems that approach the problem in a human-like way. Then the problem we have chosen is an N-Puzzle game. The variant 8-puzzle has been used to evaluate the game solver.

1.3 Aim and Objectives

1.3.1 Aim

Aim is to design and develop game player systems based on multi agent technology and expert systems that are based on human-like decision making process.

1.3.2 Objectives

- Critical review of research in game playing and multi agent systems.
- In depth study of various technologies, methodologies and theories used in multi agent game playing.
- Design and develop MAS based solution to play games.
- Evaluate proposed solutions based on performance and accuracy.

1.4 Background and Motivation

Human way of problem solving has been a focused area of artificial intelligence. Replicating our way of thinking, problem solving is still researched and experimented. Even when we play simple games the way we approach the problem is fascinating. Exploring the ways to incorporate our way of thinking when playing games into AI is the motivation for this research.

1.5 Problem in Brief

Throughout the years various AI technologies have been used in game playing. But most of the approaches still not fully represent the human way of thinking. Here we focus on applying a human way of problem solving using multi agent systems to play games.

1.6 Structure of the Thesis

This thesis has been organized as follows. In chapter 2 review of the literature has been done in areas of game playing and multi agent systems. Next in chapter 3 the technology adopted in this research has been discussed. Then in chapter 4, the approach taken for the system has been discussed. Then design of the system is discussed in chapter 5. Next in chapter 6 implementation of the system is presented. Chapter 7 presents the evaluation, experimental design and results of this research. Finally, in chapter 8 the conclusion of this research is presented.

1.7 Summary

In this chapter we discussed a brief history of the games and how it is an integral part of culture. Effect of computer systems on the evolution of new games. Advantage of developing game playing systems to other fields. After that we discussed how human thinking and decision-making process can be used to design and develop human-like game playing systems. Then we discussed the aim, objectives and brief explanation about the problem we are going to address in this project. In the next chapter we discuss the literature related to artificial intelligence technologies, and their usage to develop various game playing systems.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In the previous we presented a brief introduction, problem, aim and objective of the research. This chapter layout the literature study carried out to explore the game playing and multi agent systems. First section contains study of literature in use of various artificial intelligence technologies in the game playing area. Then follows the study of emergence of multi agent systems. After that it illustrates the application of multi agent technologies in planning, decision making, game playing areas. Finally discuss the identified research problem in this project.

2.2 Game playing using Artificial Intelligence

Game playing using artificial intelligent (AI) programs has been in the picture for a few decades now. In 1997 IBM DeepBlue [4] defeated then Chess champion Garry Kasparov. About two decades later in 2015 AlphaGo [5] by Google's DeepMind was able to defeat European Go champion by 5 to 0. For a long time, research has been done to solve specific games using AI.

Expert systems have been used to model human knowledge for decades. Game playing, one of the oldest and integral parts of human life, can also be considered as a heuristics-based task that can be performed by using expert knowledge. Many researchers are trying to develop AI game player solutions based on expert system technologies that capable of playing games like humans do [6].

Fuzzy logic is also used to develop rule-based expert systems. Micheal Aristidou and Sudipta Sarangi [7] have modeled game theories using fuzzy set theory. The solution is based on fuzzy decision theories which are introduced by Bellman and Zadeh [8].

Artificial neural networks (ANN) which is inspired from biological neural networks [9] plays a significant role in modern AI. In game playing ANN have played a significant role.

Daniel Michulke and Michael Thielscher [10] developed an artificial neural network to evaluate the state in general game playing. Their goal was to develop an ANN that can obtain evaluation functions based on game rules and states which can overcome issues of game evaluation functions that are solely based on game rules.

The development of ANN contributed to more useful deep learning technologies. These deep learning-based solutions are much more powerful than normal ANN based systems. Guillaume Lample and Devendra Singh Chaplo [11] developed a system which allows to play games using deep reinforcement learning. It was based on the LSTM model called DRQN model. They were able to obtain successful results when evaluated against humans.

Deep reinforcement learning is also used by Gudmundsson et. al [12] to develop a system to play video games. They have used data collected from popular video games and trained a convolutional neural network (CNN) using supervised learning to develop a model. The system can predict the human action given the state of the game. They evaluated the system against Monte Carlo Tree Search (MCTS) and actual humans and were able to show that it can predict accurately.

Sha'ban et. al [13] used a genetic algorithm to solve the 8-puzzle problem. They encoded the game state in a chromosome. By evaluating the system, they have shown that the solution is simple but very powerful.

Miranda et. al [14] able to develop human-like game players that can play pacman video game. This was based on the Neuro evolution technique. Here they have used machine learning and artificial neural networks-based model to generate instructions for the player.

2.3 Emergence of Multi Agent Systems

During the 80's and 90's the paradigm of AI shifted from single agent based computing to multi agent systems [15]. Emergence of multi agent systems occurred due to increase of size and complexity of previous systems when complexity of problem increased.

2.4 Game Playing using Multi Agent Systems

In recent past multi agent systems have been used to play a variety of games such as Chess [16], Go [17], No-press Diplomacy and Risk [18]. They have chosen the multi agent system due to its ability to model and solve problems through communication, coordination and negotiation [16]–[19].

Compared to other board games Go is a relatively complex game to solve using AI. Daniel Kunkle [17] has analyzed the properties of Go such as complexity, interactivity, and non-linearity and designed a multi agent based solution to play the game.

Some games such as No-press Diplomacy and Risk have high branching factors therefore it is not possible to solve them using traditional search algorithms such as minimax. It has shown that even games like No-press Diplomacy and Risk can be solved by using a multi agent system based solution [18].

Decision making techniques used in MAS plays a key role in the success of the solution. Chen et al. [20] have studied how decisions are made in other systems and produced a learning wheel in decision making to describe non learner mechanisms in MAS decision making.

Christian et al. [21] discuss the problems agents face when planning their actions due to actions of other agents. They propose a non-deterministic planner to plan actions of agents in a multi agent environment. They have demonstrated the planner using several games such as Tic-Tac-Toe and Connect4.

Ivan Babanin et al. [22] discuss how 8-Queens puzzle can be solved using multi agent technology. In that they modeled the game chess-pieces as software agents and experimented with various decision-making processes such as random next move generation, conflict solving through negotiation.

Kamel and Marwan [23] addressed the puzzle problem by using MAS technology. They have modeled the puzzle blocks as reactive agents. It shows that alternative to traditional AI technologies such as Breadth-first search and depth-first search, a comparable solution can be reached by interaction between agents.

2.5 Limitations and Current Approaches

A summary of literature study is given in Table 2.1. It shows various uses of technology to play games. This also highlights the use of different approaches to address game playing using multi agent systems. By observing many approaches in game playing using AI technologies there are multiple decision-making strategies within multi agent technology that are still open to research.

Table 2.1: Summary of the Literature Study

Research	**Problem Addressed**	**Technology**
[4] Deep Blue	Play chess game using parallel search algorithms	1. Search algorithms
[5] Mastering the game of Go with deep neural networks and tree search	Play Go game using neural network and tree search based solution	1. Search algorithms 2. Artificial Neural Networks
[7] Games in Fuzzy Environments	Fuzzy theories to develop games	1. Fuzzy theory
[10] Neural Networks for State Evaluation in General Game Playing	ANN for evaluate the game state in General game playing	1. Artificial Neural Networks
[11] Playing FPS Games with Deep Reinforcement Learning	Deep reinforcement learning for game playing	1. Deep Reinforcement Learning
[13] Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem	Solve 8 puzzle game using genetic algorithm	1. Genetic Algorithm
[14] A Neuroevolution Approach to Imitating Human-Like Play in Ms. Pac-Man Video Game	Using artificial neural networks and genetic algorithm develop human like player to play pacman like game	1. Artificial Neural Networks 2. Genetic Algorithm
[17] The Game of Go and Multiagent Systems	Analyze Go like complex problem solving using multi agent systems	1. Multi Agent Systems
[18] A Multi-Agent System for playing the board game Risk	Using multi-agent systems to solve high complexity board games	1. Multi Agent Systems
[20] Decision making model in multi-agent system	Analyzing multi agent systems with learning wheel based dynamic decision	1. Multi Agent Systems

	making mechanism	
[21] Planning for a single agent in a multi-agent environment using FOND	Addressing planning issues when multiple agents operate in a multi agent environment	1. Multi Agent Systems
[22] Multi-agent Solution for “8 Queens” Puzzle	Applying multi agent systems approach to solve 8 queens puzzle	1. Multi Agent Systems
[23] Solving the Puzzle Problem using a Multiagent Approach	Using multi agent as alterative approach to solve puzzle problem	1. Multi Agent Systems

2.6 Problem

By reviewing literature, we have identified that problem solving by multi agent systems can be modeled in a variety of ways. Also found that different kinds of negotiation and decision-making strategies can be used in agents of the multi agent system. Therefore, we identified that an approach that involves human-like decision making process by using multi agent technologies and expert systems can be used to design a game player system.

2.7 Summary

This chapter we performed a detailed study of literature in terms of artificial intelligence technologies and their use in game playing. Then we illustrated a detailed study of multi agent systems technology, how it began, current state and the usage in the game playing area. Finally, we have highlighted the problems and limitations that we identified and presented our multi agent based game playing solution. In the next chapter we discuss technology that we adopted in detail.

CHAPTER 3

TECHNOLOGY ADAPTED

3.1 Introduction

In the previous chapter we have identified the technologies to address our game playing problem. Now in this chapter we discuss the technology adopted for the implementation of game player system. Here we discuss the multi agent systems and expert systems. Then we perform a study on current solutions available in these areas.

3.2 Multi Agent Technology

Initially AI solutions based on single agent/program [15]. When the complexity and size of the problem grew, limitations of single monolithic systems started to appear. They are limited in scalability, re-usability and have high complexity of design, development and maintenance [15]. The issue of complex problems handled by adopting “divide and conquer” policy [15]. This kind of system was initially known as Distributed Artificial Intelligence (DAI), which had two subtypes: Distributed Problem Solving (DPS) and Multi Agent Systems (MAS) [24]. Multi agent system is a system consisting of interacting, autonomous, intelligent entities [19], [25] called agents. This is a relatively new area of computer science and artificial intelligence [24], [25].

3.2.1 Agent in MAS

In multi agent systems context an agent is a computer system that is in an environment and able to perform autonomous actions to achieve its objectives [25]. Simply, an agent perceives the environment through its sensors and produces actions that affect the environment. Also, agents have only limited control of its environment and not all agents have the same view of the environment. Compared to agents in other agent-based-systems, agents in MAS are intelligent, heterogeneous loosely coupled. They are not controlled by a global controller and execute asynchronously with incomplete information.

The key properties of an agent are autonomy, social ability, reactivity and proactiveness [26]. An agent is autonomous because can operate without any human intervention. Agents have social behavior because they are able to communicate with each other. Also, agents perceive the environment and react appropriately to the changes in the environment. Finally, it can initiate some action to achieve its goal.

There are several classes of agents, they are: simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents and learning agents [27].

Simple reflex agents operate only on current perception. These agents are based on simple if-then rules. For the best results agents should have a full view of the environment [27], [28].

Model-based reflex agents have some knowledge about the problem. It maintains an internal model of the world and past states. It need not have a full view of the world. Then the next action is determined by using a model maintained within the agent [27], [29].

Goal-based agents are similar to model-based agents. But it also contains knowledge about how goal can be achieved. Therefore, these agents are able to select best action out of multiple possible actions that are more favorable to achieve its goal [27], [28].

Utility-based agents are an improvement of goal-based agents. In addition to features contained in goal-based agents it has a utility function that is able to measure the quality/performance of its state [27], [29].

3.2.2 Communication in MAS

Communication between agents is extremely important for multi agent system. Having standardized communication protocols and languages improves interoperability, extendibility of multi agent systems. There are several protocols and languages such as KQML/KIF and FIPA-ACL available for communication between agents.

Knowledge Sharing Effort (KSE) is a project by DARPA to develop agent communication protocol to share knowledge between systems [30]. It consists of two

sub parts, Knowledge Query and Manipulation Language (KQML) and Knowledge Interchange Format (KIF). The KQML defines the format of the message and protocol for message handling [30]. The KIF defines the knowledge part [25], [26] of the message. Due to limitations such as lack of semantics in KQML led to development of FIPA-ACL [25].

FIPA-ACL is an agent communication language developed by the Foundation of Intelligent Physical Agents (FIPA). It defines a set of Communicative Acts or performatives to define the intent of the message [26], [31]. However, this does not define the format of the content of the message. This language is used by several agent modeling frameworks such as JADE, SPADE, etc.

3.2.3 Agent modeling software and frameworks

There are several multi agent system development frameworks available. These provide tools and methodologies to model and implement multi agent systems. Some of them are NetLogo, JADE and SPADE.

3.2.3.1 NetLogo

NetLogo is a programming language and a modeling environment to develop and simulate multi agent systems [32]. This was designed by Uri Wilensky in Northwestern University to use in research and educational purposes. It was developed using Java programming language. Having a GUI is an added advantage of NetLogo. Simplicity and user friendliness of NetLogo allows students to easily model and simulate complex systems. This is able to handle models with thousands of agents. NetLogo targets results to be scientifically reproducible therefore requires the models to be deterministic.

3.2.3.2 JADE

“JAVA Agent DEvelopment Framework” or JADE is a FIPA complaint multi agent system development framework [33], [34]. As the name implies, the framework is implemented using JAVA programming language. The goal of this project is to simplify development of multi agent systems [35]. This can be considered as a distributed middleware system that allows to implement multi agent systems. Due to

the nature of its design, JADE can be easily extended with add-ons. It contains an implementation of FIPA-ACL for the communication between agents. The framework provides run time environment and core logic that is required to implement and run standard complaint agents.

3.2.3.3 SPADE

SPADE (Smart Python Agent Development Environment) is a python based multi agent development platform [36]. It uses an XMPP based instant messaging network as the primary communication layer. It is implemented to be fully compliant in FIPA specification. It uses many features of XMPP protocol such as presence notification, multi-user conference to provide real-time state of agents and group/broadcast communication strategies, respectively. It also provides behavior strategies to use for agents.

3.3 Expert System

Multi agent systems consist of intelligent collection of agents. These intelligent agents require a knowledge source to make decisions. When humans play games, they use rules and knowledge about the game to make informed decisions.

Expert systems are able to emulate human expert's knowledge and decision-making process [37], [38]. This ability can be used to model human game players' knowledge into agents. There are many types of expert systems available such as rule-based expert systems, frame-based expert systems, fuzzy logic expert systems and neural network based expert systems [37], [38].

As games are mostly based on rules, here we use rule based expert systems to model agent's knowledge. The rules in the expert system models the human player's knowledge about the game and strategies.

3.4 Technology of Framework used in Game Player System

The multi agent framework for game player systems is based on the architecture of SPADE multi agent system. Our custom multi agent framework uses python

programming language to develop agents and expert system. The communication and storage sub system of the framework is based on Redis.

Redis is an open-source high performance in memory datastore [39]. It is capable of providing features of database, cache, and message broker. Compared to the communication system used in SPADE this is capable of providing high speed and simple solutions for agents to communicate and store data.

3.5 Summary

In this chapter we discussed the technologies adopted to implement the game player solution. Here we provided the brief introduction to multi agent systems and expert systems. Then explained about the technological solutions currently available to implement multi agent systems. After that we provided an introduction about the technology stack of a multi agent framework used to develop the game player system. In the next chapter we will discuss the approach we have selected.

CHAPTER 4

APPROACH

4.1 Introduction

In the previous chapter, based on the problem identified, we discussed the technologies adopted to develop a game player system. It contains details about the multi agent technology and expert systems. Now in this chapter we discuss the approach for the problem we identified previously in terms of hypotheses, input, output, process, users and features.

4.2 Hypothesis

The hypothesis of this research is that human-like decision making thinking process can be modeled using multi agent systems to play games. Inspiration for this hypothesis is coming from the similarity between how humans approach solving problems by thinking on different paths, weighing and arguing different solutions in their mind and how agents in a multi agent system behave by communicating, negotiating to solve problems.

4.3 Input

Inputs for the system are the state of the game at a given moment and the feedback for previous actions performed on the game by the system. State of the game can vary depending on the game. The state of the game may contain positions of characters, players, objects and information about players surrounding, etc.

4.4 Output

The output of the system will be the next action the game player should take. The action produced by the game player also depends on the type of game it is playing. For simple board game, it can be a statement like ‘move up’, ‘move down’, etc.

4.5 Process

A player of a game needs to consider many factors such as game state, previous actions when planning about the next action. Here the player will consider various

options available in the game by considering positive and negative effects in immediate and future situations.

In the proposed system, the suggested method is to use multi agent technology to process the game state to produce next action. The system composed of multiple agents, namely coordinator agent, decision agents and action agent.

The coordinator agent is responsible for communication between the game program and the game player system. It fetches the game state from the game program and send processed action back to game program.

The decision agents take the game state received from the coordinator agent as the input. The system contains multiple decision agents. Each decision agent represents a single decision/option available for the player. Then these decision agents perform analysis based on the game state and negotiate with other decision agents to analyze the effect of its decision. By negotiations and coordination between decision agents they collectively produce sequence of decisions along with utility value and sent them to action agent.

Action agent to monitor the progress of the decision agent and process the decision sequences received by the decision agents. It then performs further analysis on the decision sequences and converts the best sequence into a game-action that is understood by the game program. Then it sends the game-action to the coordinator agent to communicate to game program.

4.6 Users

Use of the technology developed for a game playing system is not restricted to specific areas or users. It can be used to solve a variety of problems in vastly different subject areas. However here we identified that this research is more useful to researchers of multi agent systems, researchers of general intelligence, developers of game and simulation applications.

4.7 Features

Using the suggested process, we are able to achieve many features. A game player system based on multi agent systems. Ability to model human decision-making process using MAS. Since agents are able to work parallelly the system is able to achieve high performance. Using the multi agent technology we can easily extend the systems with new abilities.

4.8 Summary

In this chapter we discussed the approach selected to address the problem we identified during the literature study phase. The approach of this research is presented in terms of hypotheses, input, output, process, users and features. Based on this approach, in the next chapter we discuss the design of the system in detail. It presents the overall architecture of the system, how the system takes inputs, process the inputs and then produce the output.

CHAPTER 5

DESIGN

5.1 Introduction

In this chapter the design of the game player system is presented. The design of the system is based on the approach described in the previous section. As the game player system based on multi agent technology, the architecture of the system considers the design of each agent type. Here we present the high level architecture of the system, design of individual components of the system, and high level interaction between components of the system.

5.2 High Level Architecture

The game player system will consist of three types of agents: coordinator agent, decision agent and action agent. There are multiple decision agents depending on the choices available to the player based on the game. The high level architecture of the system is given in Figure 5.1.

As shown in Figure 5.1 the game player system interacts with the game program through the coordinator agent. Coordinator agents get the game state from the game program and populate the internal environment with the retrieved game state. Upon request from the coordinator agent the decision agents will analyze the game state and produced results will be sent to the action agent for further processing. After receiving the decision messages from decision agents, the action agent converts the best decision to game-action and sends it to the coordinator agent. Coordinator agent send the received action to the game program.

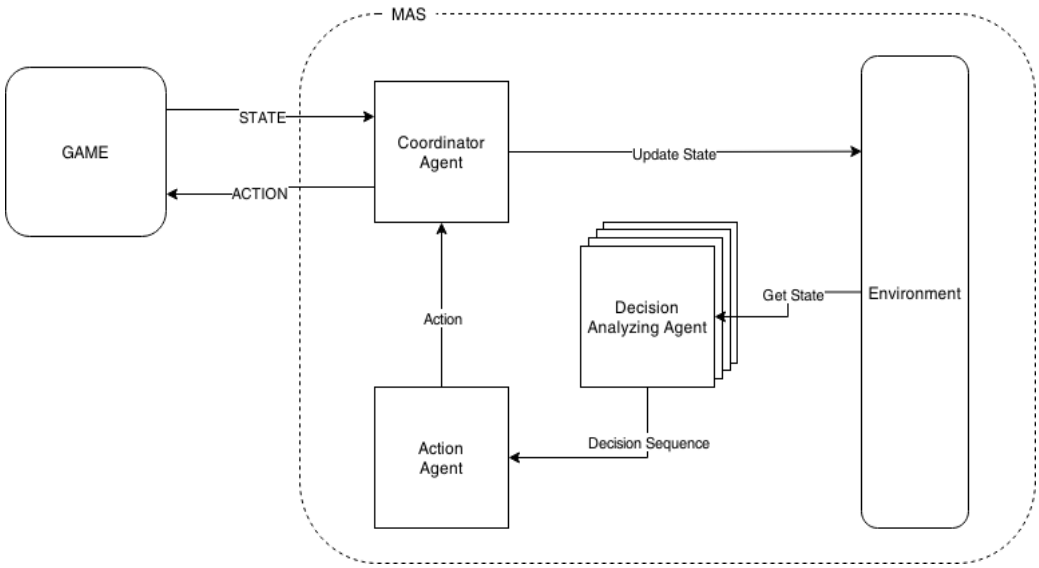

Figure 5.1: High level architecture of the multi agent game player system

5.3 Components of the MAS Game Player System

5.3.1 Coordinator Agent

Coordinator agent is the main agent in the system. Responsibilities of the coordinator agent includes fetching the game state, updating the system environment, sending the action to the game program, requesting actions from decision and action analysis agents and overall management of the system.

5.3.2 Decision Agent

Based on the available decision for a player, the system will have multiple decision agents. Each decision agent represents an option/decision available to the player. The agent takes the game state and previous action as the inputs. It uses the knowledge/rules to analyze the viability of the decision. This process is shown in Figure 5.2. By communicating with other decision agents, they generate decision sequences to analyze effects of the decision in the future.

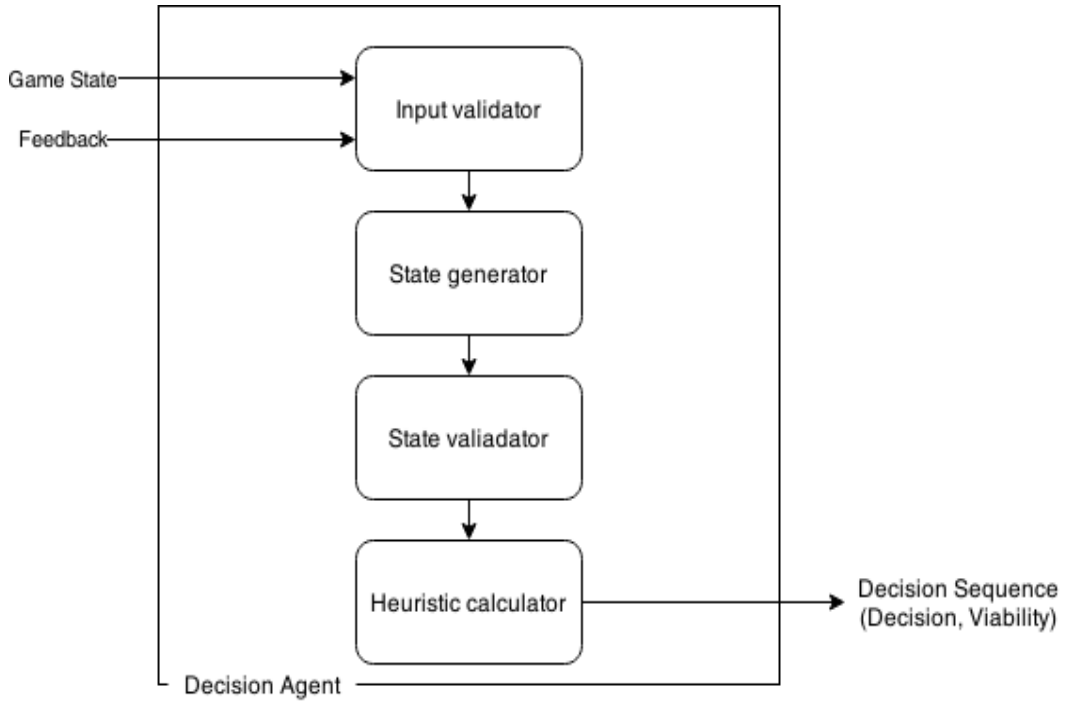

Figure 5.2: Design of Decision Agent

5.3.3 Action Agent

The action agent gathers all decision sequences and analyzes for any unforeseen side effects and selects the best decision sequence. After selecting a decision sequence, it converts the decision into action and sends it to the coordinator agent. The design of the action agent is illustrated in Figure 5.3.

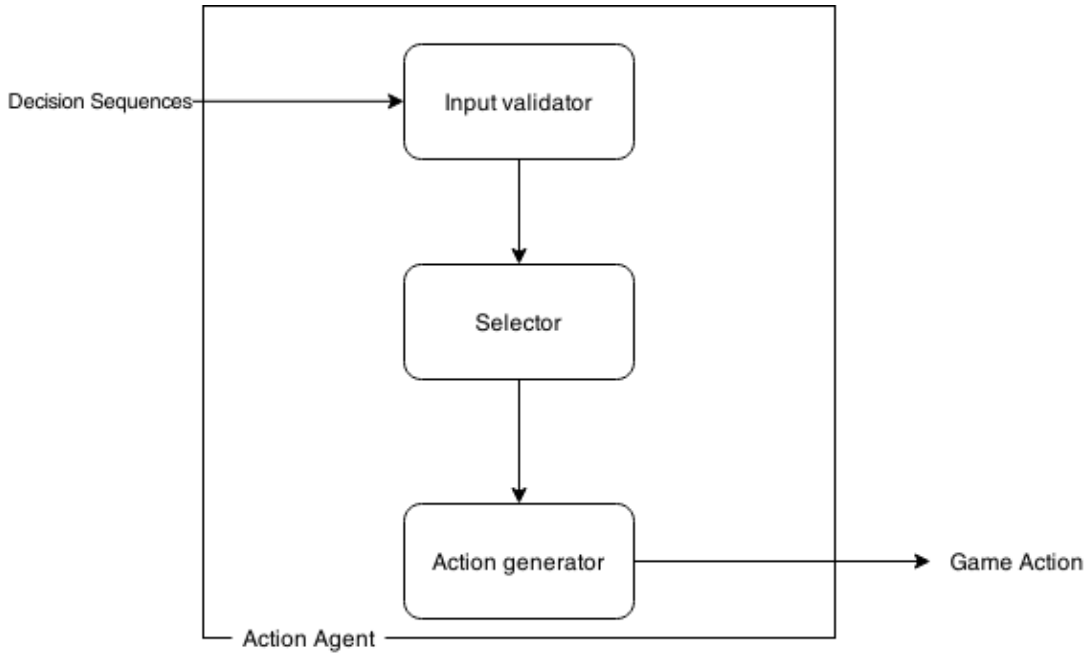

Figure 5.3: Design of action agent

5.3.4 Communication Layer

The communication layer of the system facilitates all the communication between agents. Each agent will have a private communication channel to receive private messages and a common channel to receive broadcast and multicast messages.

5.4 Summary

In this chapter the design of the game player system is discussed. The discussion includes the design of the high level architecture, design of components and interaction between components. It also discusses the design of each agent type present in the game player system. Based on this design and the technology presented in chapter 3, in the next chapter we discuss the implementation of the game player system. It contains detailed descriptions of how tools and technology solutions are used in the implementation.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This chapter discusses the implementation process of the design stated in the previous chapter. First part of this section describes the implementation of each module. Then discuss the integration of the component.

This game player system is implemented based on multi agent technology. To implement the agents and other modules of the system python programming language has been used. For implementing real time communication layer and storage layer Redis database is used. Figure 6.1 illustrates how the communication and storage layers interact with agents in the system.

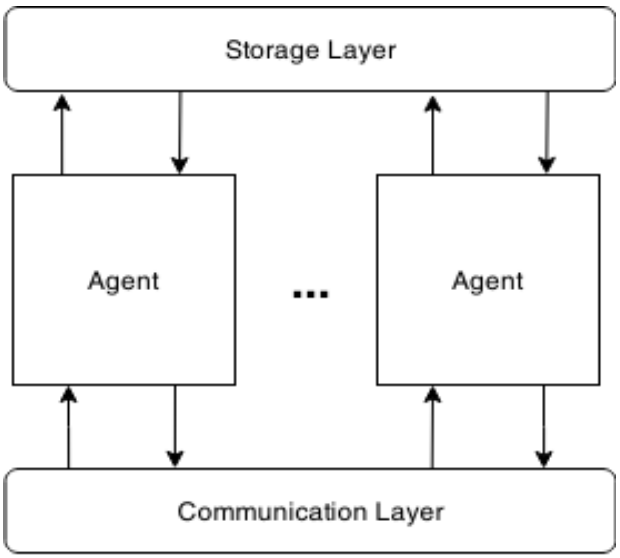

Figure 6.1: Interaction between agents, communication layer and storage

6.2 Tools and Technology

The game player system is implemented using a custom multi agent system developed especially for this project. This multi agent system is developed using python programming language and uses Redis for storage and communication.

6.2.1 Python

Python is a general purpose, high level programming language [40]. It supports multiple programming paradigms such as, imperative, functional, object-oriented and procedural. Because the language is easy to learn and provide powerful features it is popular among students, teachers, researchers and engineers. Due to the large community of programmers, Python has a vibrant ecosystem of libraries, tools and learning materials. Availability of resources and user friendliness are the main reasons multi agent framework used in this research developed using python programming language. Version 3 of the language has been used in this development process.

6.2.2 Redis

Redis is an in-memory data structure store [39]. The storage and message broker features of this system are heavily used in the development of the multi agent framework. The key-value storage system is used to implement the storage component and the message broker system has been used to implement the communication component of the multi agent framework.

6.3 Agent Implementation

Each agent in the system is implemented using python language. Agent programs run as a separate process in the OS. Therefore, the agents can be run in the same computer or in different computers. Every agent contains a communication module and storage module to provide communication and storage abilities. Depending on the type, each agent has a rule set to process inputs such as messages, game states, etc. Figure 6.2 shows basic structure of agent program.

```
incoming_message_handler(msg):
     if msg = 'control'
          process_control_message(msg)
     if msg = 'request'
          process_request_message(msg)

process_request_message()
     // execute rules, analyze input, etc

message_dispatcher(channel, msg)
     publish_message(channel, msg)

main_event_loop():
     register_handler(incoming_message_handler)
     wait_for_message()
```

Figure 6.2: Pseudo code of an agent program

6.3.1 Coordinator Agent

The coordinator agent is the main agent of the game player system. It is responsible for all the interactions between the game program and the system such as initializing all other agents, updating the game state in the internal environment. To facilitate the given responsibilities, it consists of several sub modules, mainly game API sub module, communication sub module and message processor sub module. The Figure 6.3 shows the design of the coordinator agent.

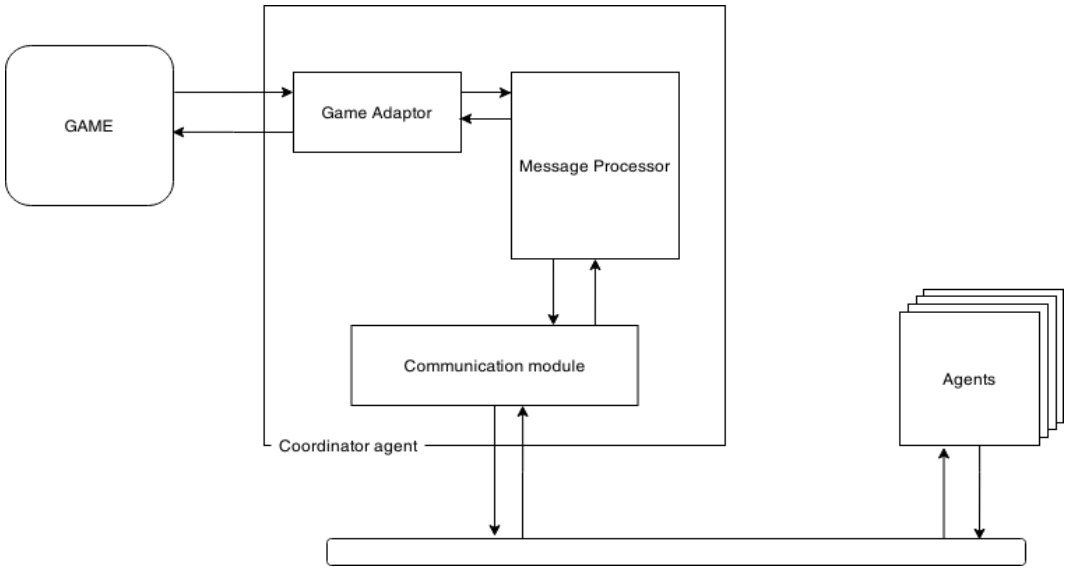

Figure 6.3: Design of coordinator agent

Message processing module handles all the incoming messages to itself. Messages can come from other agents and game program. It processes the message and dispatches to relevant sub modules within the agent. This sub module connected with the game API sub module and communication sub module within the agent.

The coordinator agent uses APIs to communicate with the game program. This part has been implemented as a separate sub module within the agent so that it can be switched or updated without affecting other parts of the agent.

Communication sub module in the agent allows the agent to connect with the communication layer of the game player system. This allows agents to communicate with other agents and the game player system. This module subscribed to coordinator private channel and common channel to receive messages.

When the coordinator agent fetches the game state and the feedback from the game program it stores the data in the environment and assigns a unique ID called game state ID. By storing in the environment, the data can be accessed by all other agents. After updating the environment, the coordinator agent sends a message to all

decision agents requesting analysis on the new game state. A sample request message is given in the Figure 6.4.

```
{
      'id': 123456,
      'type': 'need_move',
      'from': 'coordinator',
      'for_game_state_id': 4567,
}
```

Figure 6.4: Request message from coordinator agent

When the coordinator agent receives a selected action message from the action agent it sends the action to the game program through the game API. Then get the new state and feedback from the game program.

6.3.2 Decision Agents

The game player system can have multiple decision agents. This is based on the options/decisions available for the player of a game. Each decision agent represents a specific option/decision available for the player. A decision agent takes the game state and previous actions as the inputs. It uses the knowledge/rules to analyze the viability of the decision. To facilitate the required features the decision agent comprises several sub modules, communication sub module, rule engine and heuristic analyzer module.

Communication sub module facilitate the communication between other agents through the communication layer of the game player system. Decision agents subscribed to common channel and decision channel to receive messages.

Rule engine is responsible for validating the input to the agents, generating new states based on the assigned decision type and to validate the new state. These rules can be different based on the decision the agent represents. Implementation of this module is dependent on the game.

Heuristic analyzer module is responsible for anglicizing the decision taken by the agent. This will take the previous game state, generated game state and the expected game state as the input when calculating the heuristic value.

As illustrated in Figure 5.2 when a decision agent receives a request to analyze a game state, first it validates the incoming request by comparing the game state ID in the message against the current game state ID in the environment. This is to make sure that the agent is not processing an old message. After the validation it generates a new game state based on the decision the agent represents. Then the new game state is validated to check whether it is a valid state. Finally, heuristic value is calculated to measure the quality of the decision.

When analyzing a decision, it is not enough to check the viability of the given time. It also needs to analyze how the decision will affect in the future. For this purpose decision agents generate a sequence of decisions based on the first decision. This is done by sending the generated decision and its heuristic value to all the other agents. Then the other agents will consider it as a new game state and perform the decision analyzing again. This is repeatedly continued until pre-configured depth of decision sequence is generated. The Figure 6.5 shows simplified communication flow that occurred when generating a decision sequence.

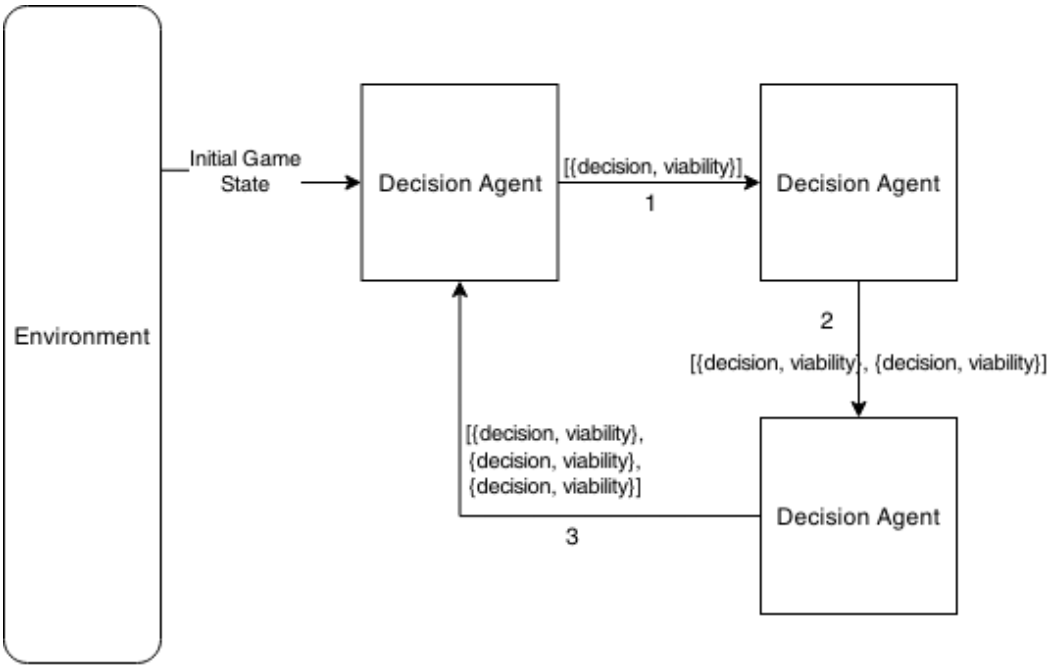

Figure 6.5: Decision sequence generation through communication between decision agents

When sufficient depth of the decision sequence is generated the final decision, agent sends the sequence as shown in Figure 6.6 to the action agent. There will be multiple decision sequence messages sent to the action agent.

```
{
     'id': 123456,
     'type': 'decision',
     'from': 'up',
     'for_game_state_id': 4567,
     'sequence':[{'a':'up','h':2},
                 {'a':'left','h':3}, ...],
}
```

Figure 6.6: Decision sequence message

6.3.3 Action agent

Action agent is responsible for gathering the decision sequences, validating the sequences for any issues, and converting the best decision sequence to a game action. The action agent contains a communication sub module, knowledge/rule engine.

Communication sub module allows the agent to communicate with other agents in the system. Only subscribed to its private channel. The decision agents send the messages to the action agent's private channel. The action agent sends the messages to the coordinator agent by publishing messages to the coordinator agent's private channel.

Rule engine in the agent is responsible for analyzing the decision sequence of any issues and to convert the sequence to game action. This is done to make sure the decision sequence is valid and to avoid any deadlock situations.

6.4 Communication Layer

The communication layer of the system facilitates communication between agents. The system uses broadcast, multicast and unicast messages transmissions to communicate between agents. All these message transmission patterns are implemented based on the pub-sub feature of Redis system.

To broadcast a message, it is published to a common channel that is subscribed by all agents. Multicast transmissions used by decision agents to communicate with each other. This is done by using a channel that is subscribed only by decision agents. Messages published to this channel only received to decision agents. To send messages only to a specific agent (unicast transmission) message is published to the agent's private channel. An agent subscribes to a relevant channel depending on the agent type when it initializes.

When an agent needs to send a message to a specific agent it publishes the message to the receiver agent's private channel. Similarly, messages published into the common channels received by all agents.

Data in the message is serialized using JSON format. Each message contains at least an ID and a TYPE field. Depending on the type of the message there can be additional data fields. A sample message is shown in Figure 6.7.

```
{
     id: 123456789,
     type: 'need_move'
     from: 'coordinator'
     ...
}
```

Figure 6.7: Agent message format

6.5 Storage Layer

The storage layer allows to store environment data and agent's private data. This is implemented using Redis key-value storage. When an agent wants to persist data, it can directly write the data into Redis storage. As we use key value pairs to store the data, each agent prefix it's identifier in the 'key' section of the data. This allows agents to store data without risk of accidental modifications.

6.6 Summary

In this chapter we discussed the implementation of the game player system based on the design provided in the previous chapter. The implementation includes the development of a multi agent framework used in the game player system along with brief description about the technology stack and each subcomponent of the game player system. The components of the game player system include coordinator agent, decision agents, action agents and communication and storage layer. The next chapter discusses how we evaluated the developed system.

CHAPTER 7

EVALUATION

7.1 Introduction

In this chapter we discuss the evaluation process of the implemented game player system. Here we start by designing an experiment to measure the time taken to play an 8-puzzle game. For the experiment we discuss how we involved human players and A* based game solver. Then we discuss the data collection process of the experiments performed. Finally, we discuss the results obtained and based on that the performance aspect of the game player system.

7.2 Experimental Design

An experiment has been designed to test the game player system using N-Puzzle. This puzzle problem can be solved using search algorithms such as Breadth First Search, Depth First Search, A* Search, etc. [41]. Here we compare MAS game player against human players and A* based solver. We collect average time taken to solve the game.

7.2.1 N-Puzzle Game

N-Puzzle is puzzle game with N number of sliding tiles and one blank spot in m * m grid within a square frame. The number of tiles N depends on the size of the grid such that N = m2 – 1 [42], [43]. Common of the variants of this puzzle are 8-puzzle and 15-puzzle where N=8 and N=15 respectively (Figure 7.1).

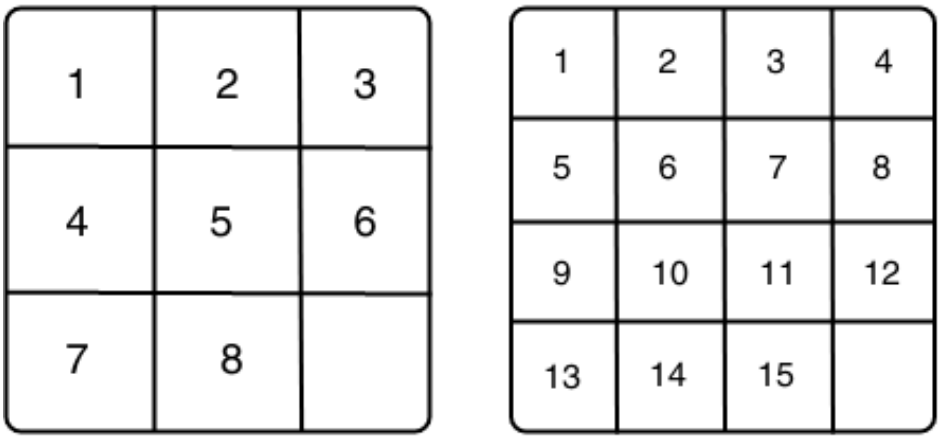

Figure 7.1: 8-Puzzle (Left) and 15-Puzzle (Right)

At the start the order of the tiles will be random. The goal is to arrange the tiles to match the goal configuration. This is done by moving the tiles near blank space to up, down, left or right. A sample initial state and goal state for 8-puzzle illustrated in Figure 7.2. Here we have chosen 8-puzzle as we are comparing against humans and it is simple for humans to solve.

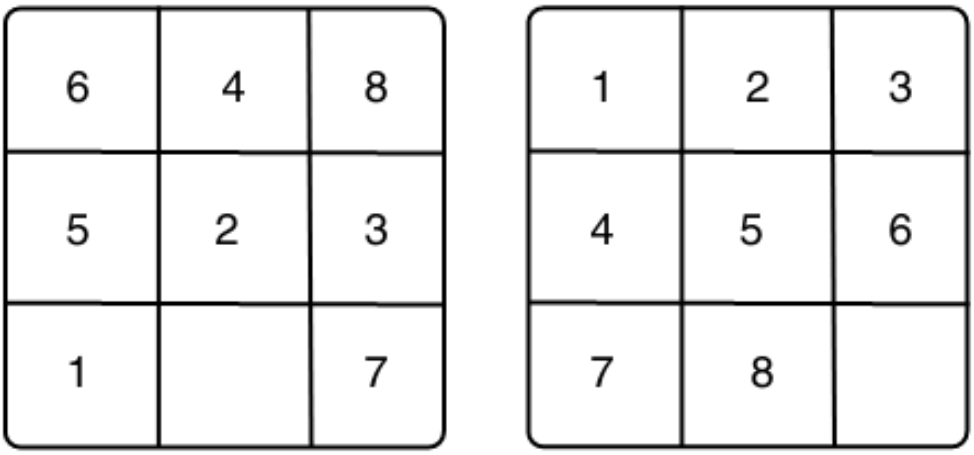

Figure 7.2: 8-Puzzle with an initial configuration (left) and goal configuration (right)

7.2.2 Human player

As the proposed solver is designed to follow the human reasoning process. It is essential to compare the multi agent game solver program against real human players. For that purpose, a group of 5 people were used. Each person was given a set of games with different starting positions. The players then record the starting time and ending time after they reach the goal for each game.

7.2.4 A* N-Puzzle Solver

A N-Puzzle solver based on the A* search algorithm will be used as the control for the experiment [41], [44]. A* is a graph traversal and path finding algorithm [45]. This uses best-first approach based on A* algorithm to traverse through the state space of the board configurations to find the path to goal state. Input for this solver will be the initial board configuration and the goal board configuration. Then by applying the A* algorithm using manhattan distance as the heuristic it will find the goal state.

7.2.5 N-Puzzle for MAS Game Player

The implementation of the game player is adopted to play N-Puzzle game. There are four variants of decision agents have been configured namely UP Agent, Down Agent, Left Agent and Right Agent.

The number of future steps to consider when generating the decision sequence is also configured to 3. Therefore, the decision (UP, DOWN, LEFT, RIGHT) agents will consider three future steps when selecting the action.

- Up Agent
    - This agent analyses the viability of moving the empty block of 8-puzzle to one position in up direction.
- Down Agent
    - This agent analyses the viability of moving the empty block of 8-puzzle to one position in down direction.

- Left Agent
    - This agent analyses the viability of moving the empty block of 8-puzzle to one position in the left direction.
- Right Agent
    - This agent analyses the viability of moving the empty block of 8-puzzle to one position in the right direction.

7.3 The Setup

The experiment is done on an 8-puzzle variant of the game. A prepared set of starting board configurations were used for each player/solver. Then the human players, A* solver and our game player has been used to play each game configuration.

The Human players and A* solver will be the control of this experiment. Input will be the initial game state. In each player's game starting configuration, time elapsed to solve the game. Fields in the sample data set are given in Table 7.1.

Table 7.1: Sample output of game solver

Configuration	**Elapsed Time (seconds)**
5, 7, 3, 1, 2, 8, 0, 6, 4	79
7, 1, 4, 6, 3, 5, 8, 2, 0	52

7.4 Evaluation Strategy

Performance of the game solver measured by time to reach goal state against the result produced by Human players and A* solver. Here the average time taken is compared.

There are several methods to evaluate the quality of the move and game state. Here we use a number of misplaced tiles and the Manhattan distance as the heuristics to measure the quality.

The utility function will be the summation of the inverse number of misplaced tiles and the Manhattan distance.

7.4.1 Heuristics

7.4.1.1 Number of Out of Place Tiles

This gives the number of tiles that are out of place compared to goal state. If all the tiles are in the correct place the count is zero. If two tiles are switched the count is 2. Lower the number quality of the state is higher.

OT = number of tiles that are out of place with respective to goal state

7.4.1.2 Total Manhattan Distance

Manhattan distance between each tile's current position and goal position is considered. The total distances give the overall status of the given state. If all tiles are in goal position, then the total distances will be zero.

MD = sum(manhattan distance of each tile compared to goal state)

7.4.2 Utility of a board state

To calculate the utility of a given n-puzzle state inverse of total out of place tiles and Manhattan distances is considered. Here higher the utility value the quality of state is better.

Utility = - (OT + MD)

7.5 Experimental Sample Data

N-Puzzle can have many starting positions. To be able to compare the results in an unbiased manner it needed to have common starting positions. Here we planned to use 25 different starting positions as the input for each game solver. In order to generate a starting board configurations board was randomly moved in random directions for a random number of steps. By repeating the generation step for all required numbers of starting positions were collected.

7.5.1 Collecting data from human players

There were 5 human players used in this test. Each player was given all 25 game configurations to solve and elapsed time was recorded. A sample format of this dataset is shown in Table 7.2.

Table 7.2: Human player time sheet

Configuration	**Elapsed Time (seconds)**
5, 7, 3, 1, 2, 8, 0, 6, 4	79.00
7, 1, 4, 6, 3, 5, 8, 2, 0	52.00

After collecting data from all players, the average time for each starting game configuration was calculated and recorded under “Human player avg. time” column of Table 7.4.

7.5.2 Collecting data from A* n-puzzle solver

Similar to human game players all the board configurations were fed into A* N-puzzle solver and time to solve the puzzle was recorded. Sample format of this was shown in Table 7.3.

Table 7.3: A* solver time sheet

Configuration	**Elapsed Time (seconds)**
5, 7, 3, 1, 2, 8, 0, 6, 4	2.58
7, 1, 4, 6, 3, 5, 8, 2, 0	20.10

To remove effects from other programs, the same configuration was executed multiple times and average time was taken. This average time then recorded under "A* solver avg. time" column in Table 7.4.

Table 7.4: Elapsed time to solve the 8-puzzle

Configuration	**Human player avg. time (seconds)**	**A* solver avg. time (seconds)**
3, 6, 7, 5, 4, 8, 0, 2, 1	51.12	9.27
2, 7, 3, 6, 1, 8, 4, 5, 0	36.82	4.77
1, 8, 2, 4, 5, 6, 7, 3, 0	26.57	2.55
6, 3, 8, 4, 1, 2, 5, 7, 0	20.26	24.47
4, 2, 3, 7, 5, 6, 1, 0, 8	33.80	3.14
8, 7, 4, 3, 6, 2, 1, 0, 5	21.33	21.02
4, 1, 0, 3, 2, 5, 8, 6, 7	11.61	10.24
8, 4, 0, 5, 3, 7, 2, 6, 1	40.51	19.25
0, 4, 7, 6, 8, 2, 5, 3, 1	19.51	20.94
6, 8, 4, 5, 3, 2, 1, 0, 7	64.81	5.55
0, 5, 1, 6, 2, 3, 7, 4, 8	25.46	17.15
6, 0, 3, 8, 4, 7, 2, 1, 5	14.09	10.89
1, 0, 3, 6, 8, 5, 7, 2, 4	23.47	9.85
8, 1, 4, 6, 7, 2, 5, 3, 0	21.05	3.78
0, 6, 2, 5, 8, 1, 4, 3, 7	17.94	16.77
6, 1, 7, 2, 4, 3, 0, 5, 8	17.03	2.81
3, 7, 5, 4, 1, 6, 8, 0, 2	36.37	2.74
4, 5, 3, 7, 1, 6, 2, 0, 8	11.16	2.09
4, 8, 5, 2, 3, 0, 6, 7, 1	64.57	28.69
1, 4, 2, 5, 7, 0, 8, 6, 3	14.42	2.01
7, 6, 8, 1, 3, 4, 0, 2, 5	31.10	9.59

6, 1, 2, 0, 7, 8, 5, 4, 3	20.41	2.68
4, 7, 1, 3, 8, 2, 0, 5, 6	23.83	3.66
1, 5, 6, 8, 0, 4, 2, 3, 7	26.60	7.19
0, 8, 4, 7, 1, 2, 3, 6, 5	18.32	10.77

7.6 Experimental Results

Same board configurations used to collect sample data were used to evaluate the game player system. Each board configuration is provided as the input and time elapsed to reach the goal state was recorded. To any outside factors from effecting the data. Each starting board configuration was executed three times and average time was calculated. Fourth column of Table 7.5 contains the average time MAS game players took to solve each configuration.

Table 7.5: Elapsed time to play 8-Puzzle for game player system

Configuration	**Human player avg. time (seconds)**	**A* solver avg. time (seconds)**	**MAS game player avg. time (seconds)**
3, 6, 7, 5, 4, 8, 0, 2, 1	51.12	9.27	20.21
2, 7, 3, 6, 1, 8, 4, 5, 0	36.82	4.77	10.01
1, 8, 2, 4, 5, 6, 7, 3, 0	26.57	2.55	15.34
6, 3, 8, 4, 1, 2, 5, 7, 0	20.26	24.47	26.04
4, 2, 3, 7, 5, 6, 1, 0, 8	33.80	3.14	16.54
8, 7, 4, 3, 6, 2, 1, 0, 5	21.33	21.02	36.19
4, 1, 0, 3, 2, 5, 8, 6, 7	11.61	10.24	17.51
8, 4, 0, 5, 3, 7, 2, 6, 1	40.51	19.25	14.58
0, 4, 7, 6, 8, 2, 5, 3, 1	19.51	20.94	26.18
6, 8, 4, 5, 3, 2, 1, 0, 7	64.81	5.55	30.87
0, 5, 1, 6, 2, 3, 7, 4, 8	25.46	17.15	20.69
6, 0, 3, 8, 4, 7, 2, 1, 5	14.09	10.89	13.58
1, 0, 3, 6, 8, 5, 7, 2, 4	23.47	9.85	18.54
8, 1, 4, 6, 7, 2, 5, 3, 0	21.05	3.78	15.71
0, 6, 2, 5, 8, 1, 4, 3, 7	17.94	16.77	20.49

6, 1, 7, 2, 4, 3, 0, 5, 8	17.03	2.81	16.02
3, 7, 5, 4, 1, 6, 8, 0, 2	36.37	2.74	24.58
4, 5, 3, 7, 1, 6, 2, 0, 8	11.16	2.09	19.21
4, 8, 5, 2, 3, 0, 6, 7, 1	64.57	28.69	20.15
1, 4, 2, 5, 7, 0, 8, 6, 3	14.42	2.01	15.24
7, 6, 8, 1, 3, 4, 0, 2, 5	31.10	9.59	15.21
6, 1, 2, 0, 7, 8, 54, , 3	20.41	2.68	20.01
4, 7, 1, 3, 8, 2, 0, 5, 6	23.83	3.66	18.25
1, 5, 6, 8, 0, 4, 2, 3, 7	26.60	7.19	21.54
0, 8, 4, 7, 1, 2, 3, 6, 5	18.32	10.77	20.15

7.7 Discussion on Results

We have designed the experiment to test playing time of each player system namely human players, A* game solver and MAS game player system. For each system collection of starting game configurations were given and elapsed time to play the game was recorded. Then the average time to play each configuration was calculated. In Table 7.6 show the average times of each player system to solve the game. It can be seen that the human players have the highest average playing time and A* solver has the lowest average playing time. Average playing time of our MAS game player system resides be human player and A* solver.

Table 7.6: Average time to play 8-puzzle for each player system

Human player avg. time (seconds)	**A* solver avg. time (seconds)**	**MAS game player avg. time (seconds)**
27.69	10.07	19.71

7.8 Summary

In this chapter we discussed the evaluation process of the implemented game player system using N-puzzle game. Here we set up an experiment to evaluate the time taken to play the game. Our game player system is compared against both human

players and A* based game solver program. By comparing the data from our game player system, human players and A* players. In many cases the system is able to outperform the human players. However, for some instances it was not able to pass A* searching algorithm. As we followed a human-like decision process, it may be the reason our game player system is not able to outperform A* search algorithm based game solver. In the next chapter we conclude our work by highlighting what we were able to achieve, limitations and future work that can be done based on this research.

CHAPTER 8

CONCLUSION & FURTHER WORK

8.1 Introduction

In previous chapters we provided an introduction to game playing, multi agent systems. Then we presented a literature study and problem that we identified during the study. After that we presented an approach, design and implementation of a system that addressed the problem we specified. Then in the previous chapter we evaluated the system we designed and implemented. In this chapter we present the conclusion of our research work. Based on the results of the previous chapter we provide concluding remarks. Then we discuss the limitations of the system and future work that can be done to improve the work done on this research.

8.2 Concluding Remarks

Games play an integral part of our human culture. It was a main socializing tool throughout human history. With the emergence of machines and later, computers we attempted to implement game players. It became more prominent when artificial intelligence became available. This is being used to test various theories in artificial intelligence.

In this research we have presented the design, implementation and evaluation of a multi agent system based game player system that follows a human-like decision making process. The system takes the game state as the input and produces the next action as the output. The game player system is implemented using python programming language.

Based on the result of the evaluation we performed on the game player system. We have identified that the system is capable of outperforming human game players by 42%. But it also showed that the system has some difficulties outperforming algorithmic game solvers such as A* based systems. This may be due to the rule based expert system and agent negotiation process we used to make a decision when making a move. As search algorithms are specifically designed to search on large

state spaces, they may have a high advantage against our system. Since our focus is to show that the decision-making process of humans can be implemented using multi agent systems, performance against algorithmic solvers is not a significant concern.

In this project we were able to achieve all the objectives. A game player system based on multi agent technology was successfully designed and developed. And the developed game player system was capable of playing games by following the human decision-making process. It is also noted that the designed system was able to outperform human players in many cases.

8.3 Limitations and Future Work

We have identified that more work can be done to improve the performance of the system and the knowledge base of the agents. Future work on negotiation, expert system and communication layer can be used to improve the speed of the system.

It is also possible to use alternate knowledge systems in the agents. More work can be done to experiment with various types of expert systems such as fuzzy based expert systems, or artificial neural network based agents.

To improve overall performance of the system more work can be done in order to improve storage, communication layers. And to improve the negotiation process of agents within the system.

8.4 Summary

This is the final chapter of this thesis. Here we provided the conclusion of our work, limitations and finally further work that can be done based on this research.

References

[1] C. Browne, ‘AI for Ancient Games’, *Künstl Intell*, vol. 34, no. 1, pp. 89–93, Mar. 2020, doi: 10.1007/s13218-019-00600-6.

[2] S. Gulia and R. Dhauta, ‘Traditional games in India:Their origin and status in progressive era’, *Int. j. physiol. nutr. phys. educ.*, pp. 1252–1254.

[3] E. Collins, A. Cox, C. Wilcock, and G. Sethu-Jones, ‘Digital Games and Mindfulness Apps: Comparison of Effects on Post Work Recovery’, *JMIR Ment Health*, vol. 6, no. 7, p. e12853, Jul. 2019, doi: 10.2196/12853.

[4] M. Campbell, A. J. Hoane, and F. Hsu, ‘Deep Blue’, *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, Jan. 2002, doi: 10.1016/S0004-3702(01)00129-1.

[5] D. Silver *et al.*, ‘Mastering the game of Go with deep neural networks and tree search’, *Nature*, vol. 529, no. 7587, Art. no. 7587, Jan. 2016, doi: 10.1038/nature16961.

[6] D. Johnson and J. Wiles, ‘Computer games with intelligence’, in *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, Dec. 2001, vol. 3, pp. 1355–1358 vol.2, doi: 10.1109/FUZZ.2001.1008909.

[7] M. Aristidou and S. Sarangi, ‘Games in Fuzzy Environments’, *Southern Economic Journal*, vol. 72, pp. 645–659, Jan. 2006, doi: 10.2307/20111838.

[8] R. E. Bellman and L. A. Zadeh, ‘Decision-Making in a Fuzzy Environment’, *Management Science*, vol. 17, no. 4, p. B-141, Dec. 1970, doi: 10.1287/mnsc.17.4.B141.

[9] W. S. McCulloch and W. Pitts, ‘A logical calculus of the ideas immanent in nervous activity’, *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[10] D. Michulke and M. Thielscher, ‘Neural Networks for State Evaluation in General Game Playing’, in *Machine Learning and Knowledge Discovery in Databases*, Berlin, Heidelberg, 2009, pp. 95–110, doi: 10.1007/978-3-642-04174-7_7.

[11] G. Lample and D. S. Chaplot, ‘Playing FPS Games with Deep Reinforcement Learning’, p. 7.

[12] S. F. Gudmundsson *et al.*, ‘Human-Like Playtesting with Deep Learning’, in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug. 2018, pp. 1–8, doi: 10.1109/CIG.2018.8490442.

[13] R. Z. Sha’ban, I. N. Alkallak, and M. M. Sulaiman, ‘Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem’, 2010, doi: 10.33899/EDUSJ.2010.58405.

[14] M. Miranda, A. A. Sánchez-Ruiz-Granados, and F. Peinado, ‘A Neuroevolution Approach to Imitating Human-Like Play in Ms. Pac-Man Video Game’, 2016.

[15] E. Alonso, *From Artificial Intelligence to Multi-Agent Systems: Some Historical and Computational Remarks*.

[16] H. Fransson, ‘AgentChess – An Agent Chess Approach’, p. 39.

[17] D. R. Kunkle, ‘The Game of Go and Multiagent Systems’. https://www.researchgate.net/publication/2880967_The_Game_of_Go_and_Multiagent_Systems (accessed Feb. 23, 2020).

[18] F. Olsson, ‘A Multi-Agent System for playing the board game Risk’, p. 51.

[19] A. Dorri, S. S. Kanhere, and R. Jurdak, ‘Multi-Agent Systems: A survey’. https://www.researchgate.net/publication/324847369_Multi-Agent_Systems_A_survey (accessed Feb. 23, 2020).
[20] J. Chen, L. Ming, L. Xiong, F. Jia, and D. Fei, ‘Decision making model in multi-agent system’, in *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Xianning, China, Apr. 2011, pp. 4012–4015, doi: 10.1109/CECNET.2011.5768163.
[21] C. Muise, P. Felli, T. Miller, A. R. Pearce, and L. Sonenberg, ‘Planning for a single agent in a multi-agent environment using FOND’, in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, New York, New York, USA, Jul. 2016, pp. 3206–3212, Accessed: Jun. 05, 2020. [Online].
[22] ‘Multi-agent Solution for “8 Queens” Puzzle:’, in *Proceedings of the 4th International Joint Conference on Computational Intelligence*, Barcelona, Spain, 2012, pp. 278–281, doi: 10.5220/0004148502780281.
[23] K. Khoualdi and M. E.-H. Mahmoud, ‘Solving the Puzzle Problem using a Multiagent Approach’, p. 5, 2011.
[24] N. R. Jennings, K. Sycara, and M. Wooldridge, ‘A Roadmap of Agent Research and Development’, *Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 7–38, Mar. 1998, doi: 10.1023/A:1010090405266.
[25] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.
[26] M. Wooldridge and N. R. Jennings, ‘Intelligent agents: theory and practice’, *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, Jun. 1995, doi: 10.1017/S0269888900008122.
[27] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.
[28] M. Glavic, *Agents and Multi-Agent Systems: A Short Introduction for Power Engineers*. 2006.
[29] Dr. R. Jamili oskouei, H. Varzeghani, and Z. Samadyar, ‘Intelligent Agents: A Comprehensive Survey’, *International Journal of Electronics Communication and Computer Engineering (2278–4209)*, vol. 5, pp. 790–798, Jun. 2014.
[30] R. Patil *et al.*, ‘The DARPA knowledge sharing effort: Progress report’, Jul. 1998.
[31] S. Poslad, ‘Specifying Protocols for Multi-Agent Systems Interaction’, *TAAS*, vol. 2, Nov. 2007, doi: 10.1145/1293731.1293735.
[32] S. Tisue and U. Wilensky, ‘NetLogo: A simple environment for modeling complexity’, Jan. 2004, pp. 16–21.
[33] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, 2007.
[34] ‘Jade Site | Java Agent DEvelopment Framework’. https://jade.tilab.com/ (accessed Nov. 08, 2020).
[35] F. Bellifemine, A. Poggi, and G. Rimassa, ‘JADE- A FIPA compliant agent framework’, p. 12.
[36] M. E. Gregori, J. P. Cámara, and G. A. Bada, ‘A jabber-based multi-agent system platform’, in *Proceedings of the fifth international joint conference on*

Autonomous agents and multiagent systems, New York, NY, USA, May 2006, pp. 1282–1284, doi: 10.1145/1160633.1160866.
[37] H. Tan, 'A brief history and technical review of the expert system research', p. 6, 2017.
[38] C. F. Tan, L. S. Wahidin, S. N. Khalil, N. Tamaldin, J. Hu, and G. W. M. Rauterberg, 'THE APPLICATION OF EXPERT SYSTEM: A REVIEW OF RESEARCH AND APPLICATIONS', vol. 11, no. 4, p. 6, 2016.
[39] 'Redis'. https://redis.io/ (accessed Nov. 18, 2020).
[40] K. R. Srinath, 'Python – The Fastest Growing Programming Language', vol. 04, no. 12, p. 4.
[41] M. Tabassum, 'Experimental Comparison of Uninformed and Heuristic AI Algorithms for N Puzzle Solution', Nov. 2013.
[42] A. Drogoul and C. Dubreuil, 'A Distributed Approach To N-Puzzle Solving', Apr. 1998.
[43] H. Bhasin and N. Singla, 'Genetic based Algorithm for N - Puzzle Problem', *IJCA*, vol. 51, no. 22, pp. 44–50, Aug. 2012, doi: 10.5120/8347-1894.
[44] J. Klimaszewski, 'The efficiency of the A* algorithm's implementations in selected programming languages', *J. Theor. Appl. Comput. Sci. (Online)*, vol. 8, pp. 63–71, 2014.
[45] P. E. Hart, N. J. Nilsson, and B. Raphael, 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths', *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.