Exploring Multi-Agent Reinforcement Learning in a Four-Player Mancala Variant

Final Project Report

Author: Medant Sharan
Student ID: K22007678
Program of Study: BSc (Hons.) Computer Science (Artificial Intelligence)
Supervisor: Frederik Mallmann-Trenn

2 April 2025

Abstract

Traditional Mancala variants have been extensively studied in two-player contexts. Research on multiplayer variants remains limited. We propose a flexible framework for a generalized four-player Mancala game with customizable parameters. This framework enables systematic investigation of how game complexity affects reinforcement learning performance. We focus on Q-learning with experience replay to establish baseline understanding. Our experiments reveal that trained agents outperform random baselines. These agents achieve win rates up to 50.4% in single-agent scenarios and 62% in dual-agent configurations. These results far exceed the theoretical baselines of 25% and 50% respectively. We identify "strategic sweet spots" where specific board configurations create optimal learning conditions. We document emergent pseudo-cooperative behaviors between independently trained agents. Through parameter studies, we demonstrate that performance varies non-monotonically with game complexity. This challenges simplistic assumptions about state space size and learning outcomes. Our research contributes to understanding how reinforcement learning adapts to varying strategic complexity in multi-agent environments.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Medant Sharan
2 April 2025

Acknowledgements

I would like to express my sincere gratitude to my supervisor for their invaluable guidance, support, and expertise throughout this project. I am also thankful to the Department of Informatics at King's College London for providing the resources and environment necessary for conducting this research. I would also like to extend my appreciation to my family and friends for their unwavering support and understanding throughout my academic journey. Finally, I would like to bow down before the Divine Mother, by Whose grace everything is possible.

Contents

Chapter 1

Introduction

Artificial intelligence has made remarkable strides in mastering complex games, demonstrating the potential of reinforcement learning techniques to develop sophisticated decision-making agents. While considerable research has focused on popular two-player board games, including generalized algorithms that provably find optimum strategies [4], there remains an opportunity to explore traditional multi-player board games that offer unique strategic challenges.

This project investigates the application of one of the more simpler reinforcement learning algorithms called Q-Learning [31] to a novel generalized four-player variant of Mancala, belonging to a family of ancient counting and strategy game. By extending beyond the well-studied two-player versions, we aim to explore the dynamics of multi-agent learning in environments with more complex interaction patterns.

1.1 Project Aims and Objectives

The primary objectives of this research are:

- To develop a flexible framework for a generalized four-player Mancala game with customizable parameters
- To implement and evaluate reinforcement learning algorithms for training agents in this multi-player environment
- To analyze how different game configurations affect learning performance and emergent strategies
- To contribute to the understanding of multi-agent reinforcement learning in traditional board games

This research deliberately focuses on Q-learning, one of the simplest reinforcement learning algorithms, to specifically study how baseline learning approaches perform across varying levels of game complexity. By keeping the algorithm constant while varying game parameters, we can isolate the effect of environmental complexity.

1.2 Scope and Contributions

This project extends the traditional two-player Mancala game to a four-player variant, creating a novel environment for multi-agent reinforcement learning research. We focus specifically on:

- Designing a parameterized game environment that allows exploration of different complexity levels
- Implementing Q-learning with experience replay for training self-playing agents
- Evaluating the performance of trained agents against random baselines
- Analyzing emergent strategies and game dynamics across various configurations

Through this work, we aim to bridge the gap between traditional game AI research and multi-agent reinforcement learning, contributing insights that may be applicable to other strategic board games and multi-agent systems.

1.3 Report Structure

The remainder of this report is organized as follows:

- **Chapter 2: Background** provides a comprehensive review of relevant literature on reinforcement learning, multi-agent RL, Mancala variants, and previous computational approaches to these games.
- **Chapter 3: Design & Specification** details the design of the game environment and agent architecture.
- **Chapter 4: Implementation** describes the technical implementation of the environment and learning algorithms.
- **Chapter 5: Legal, Social, Ethical and Professional Issues** discusses relevant considerations related to this project.

- **Chapter 6: Results/Evaluation** presents the experimental setup, methodology, and results from training agents across various game configurations.
- **Chapter 7: Conclusion and Future Work** summarizes the key contributions of this research and suggests directions for future exploration.

Chapter 2

Background

The background chapter sets our project into context by examining relevant literature and establishing the theoretical foundations for our research. We begin by exploring the history and variants of Mancala games, followed by a formalization of our problem, the algorithms we employ, metrics for evaluation, and a discussion of related work in reinforcement learning for board games.

2.1 Introduction to Mancala Games

Mancala represents one of the oldest known families of board games, with variants played across Africa, Asia, and the Caribbean for centuries. The term "Mancala" derives from the Arabic word "naqala," meaning "to move," which aptly describes the core mechanic of these games. While the exact origins are difficult to pinpoint due to limited archaeological evidence, researchers have traced its spread through human migration patterns and cultural exchanges [7].

Traditional Mancala games share common characteristics: they typically involve moving tokens (seeds, stones, or shells) around a board, capturing opponents' pieces, and accumulating tokens in designated areas. The specific rules vary significantly across cultures and regions, leading to numerous variants, including Oware (West Africa)[24], Bao (East Africa)[9], and Congklak (Southeast Asia)[9]. This diversity reflects rich cultural adaptations of the basic gameplay concept, with some versions like Bao noted for their strategic complexity. Contemporary research suggests that different versions may not be directly related to each other [8].

2.1.1 Kalah: The Modern Two-Player Variant

Among the numerous variants, the game of *Kalah* has garnered significant attention in research. Patented in the 1950s by William Julius Champion Jr. [5], Kalah was introduced as a modern adaptation that gained popularity in the Western world. In Kalah, each player controls a row of small pits and one larger reservoir. Players take turns picking up all seeds from one of their pits and distributing them counterclockwise, one in each pit, including their own reservoir but skipping their opponent's reservoir.

This version has been extensively studied and mathematically "solved," demonstrating that the first player can always win under perfect play [16].

2.1.2 Multi-Player Mancala Variants

While two-player versions have been widely studied, multi-player variants have received less attention. A notable four-player version developed by Gary MacLeod in the early 2000s has been commercialized by Square Root Games [19, 13]. This adaptation creates interesting dynamics not present in two-player versions, including the potential for implicit alliances and increased complexity in planning captures.

The gameplay of MacLeod's version is summarized as follows:

1. **Turn Order and Setup**: The youngest player begins the game. Each player has a set of small pockets and a reservoir on their side of the board. The initial number of stones in each pocket is consistent for all players.

2. **Movement of Stones**: On their turn, a player picks up all the stones from any one of their small pockets and begins distributing them counterclockwise, placing one ball in each subsequent pocket, including their own reservoir. Stones may also be placed in opponents' pockets but never in an opponent's reservoir.

3. **Free Turns**: If the last ball of a player's move lands in their own reservoir, they earn an additional turn. This allows for chaining multiple turns in a single round if executed strategically.

4. **Captures**: If a player's last ball lands in an empty pocket on their side of the board, a capture occurs. All the stones from the corresponding pockets around the board are collected and placed into the capturing player's reservoir, along with the capturing ball itself.

5. **Skipping Turns**: A player who has no stones in any of their pockets must skip their turn until they acquire stones to play. This rule ensures continuous gameplay while penalizing players who exhaust their resources prematurely.

6. **Game End and Winning Condition**: The game concludes when all the stones on the board have been played. The winner is the player with the most stones in their reservoir at the end of the game.

Though this is very useful for our purpose by effectively adapting Mancala to a four-player format, we introduce some necessary modifications to adapt it in a self-learning agent environment for research purposes. Our project proposes a more generalized approach with some modifications to these rules by building upon MacLeod's version. Details can be found in section 2.2.3.

2.2 Model

This section formalizes the problem statement for our research on reinforcement learning in a generalized four-player Mancala game.

2.2.1 Problem Definition

We model the generalized four-player Mancala game as a multi-agent reinforcement learning problem within a Markov Decision Process (MDP) framework. The problem can be defined as:

- **Input:** A parameterized four-player Mancala game environment with states $s \in S$, where each state represents a complete board configuration including the positions of all tokens and the current player's turn. S represents all possible configurations (states) of the board.

- **Output:** Trained agent policies $\pi : S \to A$ that maximize the expected cumulative reward for each agent, where A is the set of possible actions (selecting a pocket).

2.2.2 Mathematical Formulation

Formally, our four-player Mancala game is defined as a tuple (N, A, S, P, R), where:

- $N = \{1, 2, 3, 4\}$ is the set of four players.

- $A = A_1 \times A_2 \times A_3 \times A_4$ is the joint action space, where A_i represents the set of valid actions (pit selections) available to player i.
- S is the state space, representing all possible configurations of the game board.
- $P : S \times A \times S \to [0, 1]$ is the state transition function, determining the probability of transitioning from one state to another given the actions taken.
- $R : S \times A \times S \to \mathbb{R}^4$ is the reward function, providing a real-valued reward to each player based on the state transition.

Our generalized four-player Mancala is further parameterized by:

- $p \in \mathbb{N}$: The number of pockets per player
- $b \in \mathbb{N}$: The initial number of stones per pocket

The state space size grows exponentially with these parameters:

$$|S| \approx O((p \cdot b)^{4p}) \tag{2.1}$$

This large state space motivates our choice of a modified Q-learning algorithm to ensure the reinforced AI-agents perform well while dealing with large state spaces.

(For a derivation of the approximation above along with the modifications in our Q-learning algorithm, refer to Appendix A.1.)

2.2.3 Terminology and Game Mechanics

For consistency throughout this project, we use the following terminology:

- The term ***stones*** refers to the playable entities on the board with which the players interact.
- The term ***pits*** refers to the positions on the board in which stones can be placed.
- The term ***reservoirs*** is used to refer to special pits where stones, once placed, cannot be used in play anymore. These are also used to determine the winner at the end of the game.

The game follows these core mechanics:

1. **Turn Order:** Players take turns in a fixed clockwise order. First player is chosen randomly.

2. **Action Selection:** On their turn, a player selects one of their non-empty pits.

3. **Stone Distribution:** All stones from the selected pit are distributed counterclockwise, one per pit, skipping opponents' reservoirs.

4. **Special Rules:**

   - If the last stone lands in the player's reservoir, they get an additional turn.
   - If the last stone lands in an empty pit on the player's side, they capture stones from the opposite pit. A capture means the single stone in the empty pit along with the stones in the opposite player's corresponding pit go to the first player's reservoir.

5. **Game Termination:** The game ends when any one player has no stones left in their pits.

In our version, we explore a modification to the capture rule. While in MacLeod's version a player captures from all corresponding pits around the board, in our modified version, only the stones from the opposite player's corresponding pit (1-3 are opposite players as are 2-4) to the attacking player are captured. This alteration prolongs the game, but also introduces the possibility of emergent teaming strategies among players. The modified game termination condition ensures that the games are not too long.

2.2.4 Gameplay Examples

To illustrate the gameplay of our generalized 4-Player Mancala, we consider a scenario with the initial conditions $p = 3$ (number of *pits* per player) and $b = 3$ (number of *stones* in each pit). The initial board configuration is shown in Figure 2.1.

Players are represented by different colors: green (**G**), blue (**B**), red (**R**), and yellow (**Y**). Each player's pits are numbered from 1 to p (e.g., G_1, G_2, G_3 for Green's pits). The numbers within the pits denote the numbers of stones in that pit. The four reservoirs (Res_G, Res_B, Res_R, Res_Y) are highlighted with a thicker outline.

Let us assume it is Green's (G's) turn first. G can choose one of three actions, corresponding to the three non-empty pits (G_1, G_2, G_3):

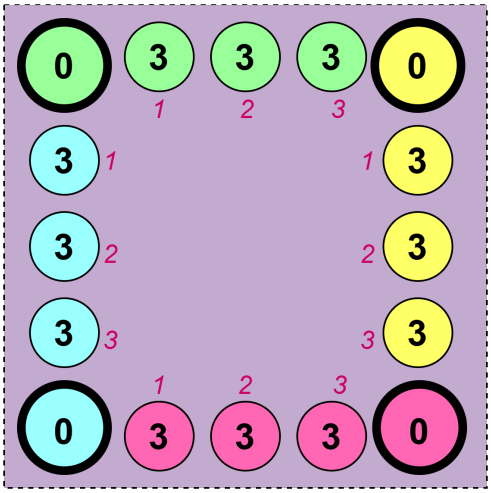

Figure 2.1: Initial board setup with $p = 3$, $b = 3$ for each pit.

- If G chooses to play pit G_1, the board transitions to the state shown in Figure 2.2. This is because G takes the 3 stones in G_1 and distributes them in the adjacent three pits (including the reservoir) counterclockwise. This would correspond to Res_G, B_1, B_2 as shown in the figure. Note that the players can only put stones into their own reservoir, if they come across another player's reservoir while distributing stones then they must skip it.

- Alternatively, if G chooses to play pit G_3, the board transitions to the state shown in Figure 2.3. In this case, the last stone lands in G's reservoir. According to the rules, G earns another turn.

Now consider another point in the game where the board looks like Figure 2.4, and it is Red's (R's) turn.

If R chooses to play pit R_3, the board transitions to the state shown in Figure 2.5. The last stone lands in an empty pit on R's side (R_1), triggering a capture.

Under the standard capture rule, R would collect stones from the corresponding pits of all other players, i.e., G_1, B_1, Y_1, adding a total of 10 stones to R's reservoir. However, using the modified capture rule, R only collects stones from G's corresponding pit G_1, adding 4 stones to the reservoir. This distinction emphasizes how the modified rule alters the gameplay dynamics.

This model provides a well-defined framework for implementing and evaluating reinforcement learning algorithms in a multi-agent adversarial setting.

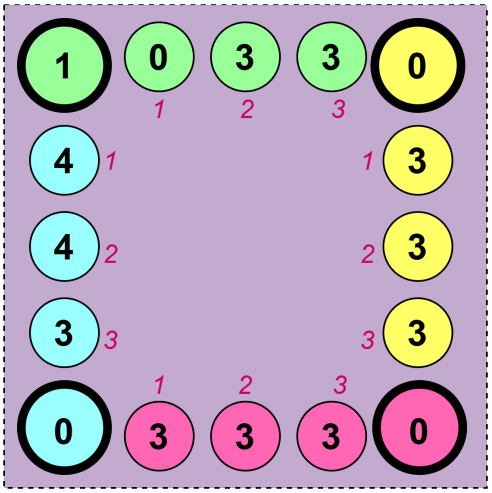

Figure 2.2: Board state after Green plays pit G_1.

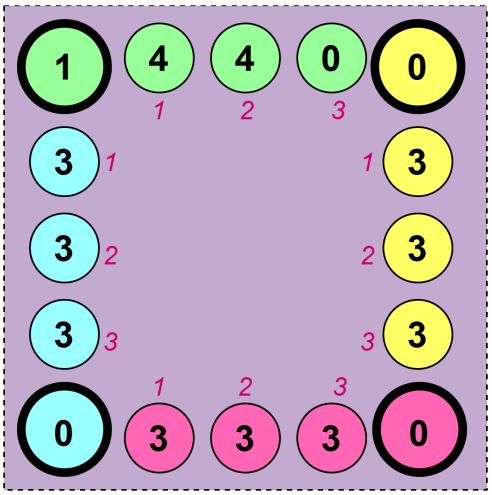

Figure 2.3: Board state after Green plays pit G_3, resulting in a free turn.

2.3 Algorithms

This section describes the reinforcement learning algorithms we employ to address the multi-agent learning problem in our four-player Mancala environment.

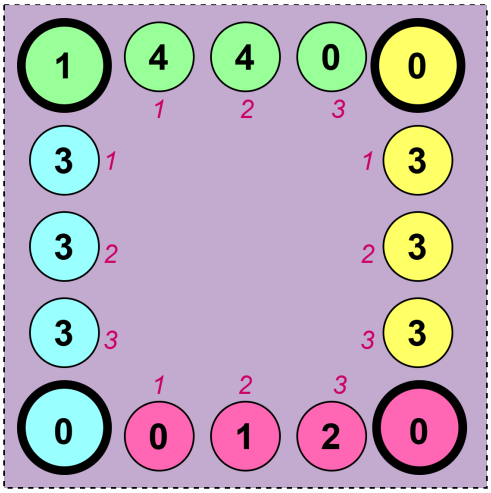

Figure 2.4: Mid-game board configuration.

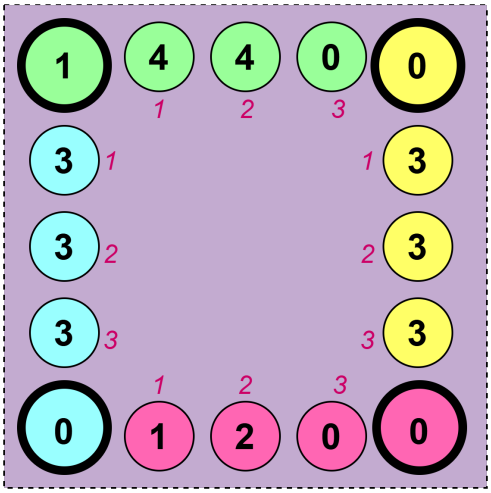

Figure 2.5: Board state after Red plays pit R_3, resulting in a capture.

2.3.1 Q-Learning

Our primary algorithm is Q-learning [32], a value-based reinforcement learning method that learns an optimal action-selection policy by estimating the expected utility of actions in states. The core update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.2}$$

Where:

- $Q(s_t, a_t)$ is the current estimate of the Q-value for state s_t and action a_t
- α is the learning rate, controlling how quickly new information overrides old information
- r_{t+1} is the immediate reward received after taking action a_t in state s_t
- γ is the discount factor, determining the importance of future rewards
- $\max_a Q(s_{t+1}, a)$ is the maximum Q-value for the next state, representing the expected future reward

While more sophisticated algorithms exist [17], we deliberately chose Q-learning for this study to examine how a fundamental, well-understood reinforcement learning approach performs when faced with varying levels of game complexity. This choice allows us to have important baseline understanding of the relationship between game parameters and learning outcomes before introducing the additional variables of more complex algorithms.

2.3.2 Experience Replay

To enhance the efficiency of Q-learning, we implement experience replay [18, 20], which stores and reuses past experiences. This approach helps break the temporal correlation in the data and enables more efficient learning by revisiting important experiences multiple times. We selected experience replay for its demonstrated effectiveness in stabilizing learning in environments with large state spaces [20].

2.3.3 Multi-Agent Learning Approach

For our multi-agent setting, we employ independent Q-learning [25], where each agent maintains its own Q-function and learns independently of the others. While this approach does not explicitly model the non-stationarity introduced by other learning agents, it provides a simple and effective baseline that can work well in practice.

We chose independent Q-learning for its simplicity and scalability to multiple agents, while acknowledging its limitations in capturing strategic interactions between agents. This approach serves as a foundation that can be extended with more sophisticated multi-agent algorithms in future work.

2.4 Metrics

This section describes the metrics we use to evaluate the performance of our reinforcement learning agents in the four-player Mancala environment.

2.4.1 Performance Metrics

We employ the following metrics to assess agent performance:

- **Score:** The final number of stones accumulated in a player's reservoir (scoring pit) at the end of the game. The player with the highest score at the end of the game is the winner.
- **Win Rate:** The percentage of games won by each agent, defined as:

$$\text{Win Rate}_i = \frac{\text{Number of games won by agent } i}{\text{Total number of games}} \times 100\% \tag{2.3}$$

- **Average Reward:** The mean reward obtained by each agent per episode:

$$\text{Average Reward}_i = \frac{1}{N} \sum_{j=1}^{N} R_{i,j} \tag{2.4}$$

  where $R_{i,j}$ is the total reward obtained by agent i in episode j, and N is the number of episodes.

- **Stones Captured:** The average number of stones captured from opponents' pits per game:

$$\text{Stones Captured}_i = \frac{1}{N} \sum_{j=1}^{N} C_{i,j} \tag{2.5}$$

  where $C_{i,j}$ is the number of stones captured by agent i in episode j.

- **Game Duration:** The average number of turns per game:

$$\text{Game Duration} = \frac{1}{N} \sum_{j=1}^{N} T_j \tag{2.6}$$

  where T_j is the number of turns in episode j.

To evaluate the performance of our trained agents, we compare them with a Random agent baseline.

2.4.2 Learning Efficiency Metrics

To assess the efficiency of the learning process, we measure:

- **Learning Curve:** The progression of win rate and average reward over training episodes.
- **Convergence Time:** The number of episodes required to reach a stable policy, defined as the point where the moving average of the win rate changes by less than a threshold ε over a window of W episodes.
- **Exploration-Exploitation Balance:** We track the exploration rate (ε in ε-greedy policy) over time to analyze how agents transition from exploration to exploitation.

2.5 Related Work

This section presents a critical evaluation of existing literature related to our research, focusing on reinforcement learning in board games, particularly Mancala variants, and multi-agent learning approaches.

2.5.1 Computational Approaches to Mancala

Early computational approaches to Mancala employed traditional AI techniques. Romein and Bal [21] developed a parallel search algorithm to solve Awari (a Mancala variant). Irving et al. [16] famously applied retrograde analysis to solve Kalah, showing that the first player always has a winning strategy in the standard configuration. This has also been mentioned previously in this chapter.

Several researchers have explored heuristic-based approaches for Mancala:

- Gifford et al. [14] provided a comprehensive evaluation of different heuristics for Kalah, comparing their effectiveness across various game configurations.
- Daoud et al. [6] employed genetic algorithms to evolve evaluation functions for Kalah, achieving strong performance against hand-crafted heuristics.

More recently, machine learning approaches have been applied to Mancala:

- T.J. Hunter [15] used six different contemporary AI algorithms to make a competitive AI-agent for classic two-player Mancala.
- Many publicly available open-source Mancala AIs are now available that are proficient in different Mancala variants, like Oware Abapa [23] and Kalah [33]

2.5.2 Reinforcement Learning in Board Games

The application of reinforcement learning to board games has a rich history, with several milestone achievements:

- Tesauro's TD-Gammon [26] demonstrated the effectiveness of temporal difference learning in backgammon, achieving expert-level play through self-play.
- Silver et al. [22] developed AlphaZero, which combined deep neural networks with Monte Carlo Tree Search to master chess, shogi, and Go without human knowledge beyond the rules.
- Mnih et al. [20] introduced Deep Q-Networks (DQN), which successfully learned to play Atari games from raw pixel inputs, establishing a foundation for deep reinforcement learning in games.

These works inform our approach to the four-player Mancala environment, particularly in understanding how agents learn in competitive and potentially cooperative multi-agent settings.

2.5.3 Research Gap and Our Contribution

While reinforcement learning has been applied successfully to various board games, and some work has explored computational approaches to two-player Mancala variants, there remains a significant gap in research on:

- Multi-player Mancala variants and their computational properties
- The effect of varying game parameters (number of pits, initial stones) on learning performance
- The application of modern reinforcement learning techniques to Mancala in multi-agent settings
- The analysis of emergent strategies in four-player adversarial board games

Our research addresses these gaps by introducing a generalized four-player Mancala framework, applying Q-learning with experience replay to this domain, and systematically analyzing the impact of game parameters on learning outcomes and strategic play.

2.6 Summary

This background chapter has established the theoretical foundations, algorithmic approaches, evaluation metrics, and related work for our research on reinforcement learning in a generalized four-player Mancala game. We have identified a significant gap in the literature regarding multi-player Mancala variants and their computational properties which we shall address.

By formalizing the problem as a multi-agent reinforcement learning task and selecting appropriate tools, we have laid a solid groundwork for our investigation into how agents learn and develop strategies in this complex adversarial environment. The following chapters will detail our implementation, results, and analysis of this approach.

Chapter 3

Design & Specification

This chapter outlines the experimental design and methodology used to investigate reinforcement learning agents in our customized novel multi-player Mancala environment. We discuss the conceptual framework, experimental parameters, and methodological approaches used to evaluate agent performance across different game configurations and positional scenarios.

3.1 Experimental Framework

Our experimental framework is designed to systematically evaluate how reinforcement learning agents perform in a four-player Mancala environment. The primary goal is to understand how game parameters affect learning efficiency and how positional dynamics influence strategic advantages. The framework is structured around three key experimental components:

1. **Parameter Studies**: Investigating how varying board configurations (pit count and stone count) affect agent learning and performance.

2. **Positioning Experiments**: Analyzing how an agent's position in the turn order affects its performance against random agents.

3. **Dual Agent Experiments**: Examining cooperative and competitive dynamics when two trained agents compete against two random agents in various positional arrangements.

This framework allows us to isolate and analyze different aspects of the multi-agent learning problem, from basic game dynamics to strategic positioning considerations.

3.2 Environment Specification

3.2.1 Four-Player Mancala Rules

The Mancala variant implemented in this research extends the traditional two-player game to accommodate four players. The core rules are as follows:

- Each player controls a section of the board with N pits, initially containing M stones each.
- The board contains a reservoir (or Mancala) for each player at the end of their section.
- Players take turns counter-clockwise, placing stones one by one around the board.
- Players skip opponents' reservoirs when olacing stones.
- If the last stone lands in the player's own reservoir, they get an extra turn.
- If the last stone lands in an empty pit on the player's side, they capture that stone along with any stones in the directly opposite pit.
- The game ends when any player's pits are empty.
- At game end, remaining stones in each player's pits are moved to their respective reservoirs.
- The player with the most stones in their reservoir wins.

This rule set maintains the strategic depth of traditional Mancala while introducing the additional complexity of multi-player interactions and positioning dynamics.

3.2.2 State and Action Representation

The environment is represented as follows:

- **State space**: The game state is represented as a vector of length $(N + 1) \times 4$, where N is the number of pits per player, and the additional space for each player represents their reservoir. Each element in the vector contains the number of stones in the corresponding pit or reservoir.

- **Action space**: For each player, the action space consists of integers from 0 to $N - 1$, representing the selection of one of their pits. Only non-empty pits constitute valid actions.
- **Reward structure**: The reward mechanism is designed to reinforce strategic play:
  - +0.5 points for each stone moved to the player's reservoir during normal play
  - +2.0 points for each stone captured from opponents
  - +3.0 points for earning an extra turn
  - Additional rewards at game end based on final score and relative position

This representation provides a comprehensive framework for agents to learn the complex strategic elements of the game while balancing immediate tactical gains with long-term positional advantages.

3.3 Agent Design

3.3.1 Learning Algorithm

We implement a Q-learning agent with experience replay to handle the large state space efficiently. The agent uses the following learning parameters:

- Learning rate (α): 00.1
- Discount factor (γ): 00.95
- Initial exploration rate (ϵ): 01.0
- Minimum exploration rate: 00.01
- Exploration decay rate: 00.9995
- Experience replay buffer capacity: 10,000 transitions
- Batch size for replay updates: 32 transitions

These values were chosen since they are quite standard for reinforcement learning experiments relating to relatively simpler board games [3, 1]. The Q-values are stored in a table indexed by state-action pairs, where states are represented as tuples of the board configuration. For efficient exploration, we employ an epsilon-greedy policy with a gradually decaying exploration rate.

3.3.2 Random Baseline Agent

For comparison purposes, we implement a random agent that selects actions uniformly from the set of valid moves. This agent serves as a baseline to evaluate the performance improvements gained through reinforcement learning.

3.4 Experimental Design

3.4.1 Parameter Study

The parameter study is designed to evaluate how different board configurations affect agent learning and performance. We systematically vary two key parameters:

- **Pits per player** (N): $\{3, 4, 6, 8\}$
- **Initial stones per pit** (M): $\{2, 3, 4, 6\}$

This creates a grid of 16 distinct configurations, each representing a different level of game complexity. For each configuration, we:

1. Train four agents playing against each other for 3,000 episodes
2. Identify the best-performing agent based on win rate during training
3. Evaluate the best agent against three random agents over 500 games
4. Calculate performance metrics including win rate, average score, stones captured, and game duration
5. Perform statistical analysis to determine the significance of performance differences

This approach allows us to identify patterns in how board parameters affect agent performance and learning efficiency.

3.4.2 Dual Agent Experiments

The dual agent experiments investigate how pairs of trained agents perform when placed in different positional arrangements. We consider six scenarios:

1. Players 1 & 2 (adjacent positioning)
2. Players 1 & 3 (opposite positioning)

3. Players 2 & 3 (adjacent positioning)
4. Players 2 & 4 (opposite positioning)
5. Players 3 & 4 (adjacent positioning)
6. Players 1 & 4 (opposite positioning)

For each scenario, we:

1. Place two trained agents in the specified positions
2. Fill the remaining positions with random agents
3. Run 500 evaluation games
4. Measure the combined and individual performance of the trained agents
5. Analyze competitive and cooperative dynamics between trained agents
6. Perform statistical analysis to identify significant patterns in positional arrangements

These experiments provide insights into how trained agents interact with each other and how their relative positioning affects their ability to exploit random opponents.

3.5 Data Collection and Metrics

Throughout the experiments, we collect comprehensive performance metrics to evaluate agent behavior:

- **Win rate**: Percentage of games won by each agent
- **Score differential**: Difference between agent scores and random baseline scores
- **Stones captured**: Average number of opponent stones captured per game
- **Extra turns earned**: Average number of extra turns acquired per game
- **Game duration**: Average number of steps per game

- **Learning curves**: Evolution of win rates during training
- **Convergence speed**: Episodes required to reach stable performance

For all metrics, we calculate confidence intervals and perform statistical significance testing to ensure the reliability of our findings. The collected data enables multi-dimensional analysis of agent performance across different experimental conditions.

3.6 Expected Outcomes

Based on the experimental design, we anticipate several key insights:

- How board complexity (number of pits and stones) affects learning efficiency and final performance
- Whether certain positions in the turn order provide strategic advantages
- How trained agents interact when placed in adjacent versus opposite positions
- The relative importance of capturing stones, earning extra turns, and other strategic elements
- Whether there are optimal configurations for agent learning in multi-player settings

Chapter 4

Implementation

This chapter details the technical implementation of our multi-player Mancala reinforcement learning framework, including the environment design, agent architecture, and experimental pipeline.

4.1 Environment Design

4.1.1 Gymnasium Framework Integration

We implemented our 4-player Mancala environment by extending the OpenAI Gymnasium [11] framework, which provides standardized interfaces for reinforcement learning environments. The Gymnasium API [12] was selected for its well-defined interaction protocols, compatibility with common RL algorithms, and established conventions for observation spaces, action spaces, and reward functions.

Our implementation inherits from the base `gym.Env` class and implements all required methods:

- `__init__()` - Initializes the environment with configurable parameters
- `reset()` - Resets the environment to initial state
- `step()` - Executes an action and returns the next state, reward, and termination information
- `render()` - Provides visualization of the environment (not implemented in our case)

4.1.2 Mancala4PlayerEnv Class

The core of our implementation is the `Mancala4PlayerEnv` class, which implements a 4-player variant of the Mancala game with configurable board parameters. The environment supports:

- Variable board configurations (adjustable number of pits per player and initial stones per pit)
- Complete game mechanics including captures, extra turns, and game termination conditions
- Reward shaping to guide agent learning
- Observation and action spaces compatible with standard RL algorithms

The observation space is defined as a Box space representing the entire board state:

$$\mathcal{O} = \{0, 1, 2, ..., M\}^{(P+1)\times N} \tag{4.1}$$

Where M is the maximum possible stones in a pit, P is the number of pits per player, and N is the number of players (4 in our implementation). The action space is a Discrete space representing the choice of pit:

$$\mathcal{A} = \{0, 1, 2, ..., P-1\} \tag{4.2}$$

4.1.3 State Representation

The game state is represented as a 1D numpy array where each index corresponds to a specific pit or store on the board. The layout follows a counter-clockwise arrangement:

- Player 1's pits: indices 0 to $P-1$
- Player 1's store: index P
- Player 2's pits: indices $P+1$ to $2P$
- Player 2's store: index $2P+1$
- And so on for players 3 and 4

This representation facilitates efficient state updates and allows agents to reason about the board configuration. The mapping between physical board positions and array indices is handled internally by helper methods that calculate valid moves and opposite pits for captures.

4.1.4 Game Mechanics Implementation

The core gameplay mechanics are implemented in the `step()` method, which processes each move according to the rules of Mancala. The implementation handles:

- Stone distribution - Distributing stones counter-clockwise, skipping opponents' stores
- Capturing - When the last stone lands in an empty pit on the player's side
- Extra turns - When the last stone lands in the player's store
- Game termination - When any player's pits are empty
- Final scoring - Moving remaining stones to appropriate stores when the game ends

To support these mechanics, we maintain several internal data structures:

- A mapping from player index to store indices
- A mapping of pits to their opposite pits for handling captures
- A tracking system for game state metadata (captures, extra turns, etc.)

4.1.5 Reward Function Design

The reward function was carefully designed to guide learning toward successful strategies. The reward calculation in `_calculate_reward()` assigns different values to various game events:

- Base rewards for stones moved to the player's store (0.5 per stone)
- Higher rewards for captures (2.0 per captured stone)
- Bonus rewards for earning extra turns (3.0 per extra turn)
- End-game rewards based on final position (25.0 for winning, 10.0 for above-average performance)

This reward structure encourages agents to develop strategies that prioritize captures and earning extra turns, while maintaining awareness of the overall game objective.

4.2 Agent Architecture

4.2.1 Q-Learning Framework

Our implementation uses tabular Q-learning [31] with experience replay for the agent learning algorithm. The `MancalaAgent` class implements an agent that:

- Maintains a Q-table mapping state-action pairs to expected rewards
- Uses epsilon-greedy exploration
- Employs experience replay to improve sample efficiency
- Tracks various performance metrics during training

The Q-learning update rule follows the standard formula [31]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{4.3}$$

Where α is the learning rate, γ is the discount factor, r is the reward, s is the current state, a is the action, s' is the next state, and a' is the next action.

4.2.2 Experience Replay

To improve learning stability and efficiency, we implemented an experience replay buffer. The `ReplayBuffer` class stores transitions (state, action, reward, next_state, valid_actions, valid_next_actions) and provides functionality to sample random batches for learning updates.

The experience replay approach offers several advantages:

- Breaks correlation between consecutive samples
- Improves data efficiency by reusing experiences
- Stabilizes the learning process

The buffer is implemented as a fixed-size deque that automatically discards oldest experiences when capacity is reached.

4.2.3 State-Action Representation

Due to the large state space of the Mancala game, we use a sparse representation for the Q-table. States are stored as tuples (converted from numpy arrays) to serve as dictionary keys, and each state entry maps to a nested dictionary of actions and their Q-values.

This approach allows the agent to only store Q-values for state-action pairs it has encountered, rather than allocating memory for the entire state-action space. New state-action pairs are initialized with small random values to encourage exploration.

4.2.4 Action Selection Strategy

The agent implements an epsilon-greedy policy for action selection in the `choose_action()` method. With probability ϵ, the agent selects a random action from the valid actions. Otherwise, it selects the action with the highest Q-value.

To handle ties in Q-values, the agent randomly selects among actions with the maximum Q-value, which helps prevent getting stuck in suboptimal policies. The exploration rate ϵ decays over time according to:

$$\epsilon \leftarrow \max(\epsilon_{\min}, \epsilon \times \epsilon_{\text{decay}}) \tag{4.4}$$

4.3 Training Pipeline

4.3.1 Training Process

The training process is implemented in the `train_agents()` function, which coordinates the interaction between multiple agents and the environment. The function handles:

- Environment initialization with specified parameters
- Training of multiple agents simultaneously
- Collection and tracking of performance metrics
- Controlled decay of exploration rates

The training loop follows the standard reinforcement learning pattern:

1. Reset the environment

2. For each step until episode termination:
   (a) Current player selects an action
   (b) Environment processes the action and returns the next state, reward, and termination information
   (c) Agent updates its Q-values based on the transition
   (d) Proceed to the next player if not terminated
3. Update performance metrics and decay exploration rates

4.3.2 Evaluation Framework

A separate evaluation framework is implemented in the `evaluate_against_random()` function, which tests the best trained agents against random agents. This evaluation:

- Measures win rates against random opponents
- Tracks score differences, stones captured, and other metrics
- Computes statistical significance of performance improvements
- Calculates confidence intervals for reported metrics

The evaluation process uses a fixed number of games with random opponents to establish a performance baseline. Statistical significance is assessed using binomial tests for win rates and t-tests for score differences.

4.3.3 Parameter Study

A key component of our implementation is the comprehensive parameter study framework. The `parameter_study()` function systematically explores different board configurations by varying:

- Number of pits per player (3, 4, 6, and 8)
- Initial stones per pit (2, 3, 4, and 6)

For each configuration, agents are trained and evaluated, and the results are analyzed for patterns in performance. The parameter study is designed to explore how board geometry affects learning and game dynamics.

4.3.4 Position Analysis

We further extended the evaluation framework to analyze the effect of player positioning. The functions `evaluate_agent_positioning()` and `evaluate_dual_agent_positioning()` test scenarios where:

- Trained agents play in different positions against random opponents
- Multiple trained agents play in various configurations (adjacent or opposite positions)

These analyses help understand positional advantages and strategic interactions between players in the 4-player game.

4.3.5 Visualization Pipeline

To support analysis and interpretation of results, we implemented a comprehensive visualization pipeline. This includes:

- Heatmaps for parameter studies
- Learning curve visualizations
- Statistical summary tables
- Comparative visualizations across configurations

The visualization functions handle data processing, statistical analysis, and generation of publication-quality figures with appropriate annotations and significance indicators.

4.4 Experimental Infrastructure

4.4.1 Experiment Management

The experimental infrastructure is designed to support reproducible research, with:

- Fixed random seeds for reproducibility
- Automatic saving of results
- Error-resilient data processing

- Progress tracking and reporting

Results from all experiments are saved as pickle files and CSV tables, allowing for subsequent analysis without rerunning computationally expensive training.

4.4.2 Statistical Analysis

Our framework includes robust statistical analysis to ensure scientific validity of results. For each metric, we compute:

- Mean and standard deviation
- 95% confidence intervals
- Statistical significance using appropriate tests

These statistical tools help identify meaningful patterns and distinguish them from random variations in performance.

4.5 Summary

The implementation described in this chapter provides a flexible framework for exploring multi-player reinforcement learning in the Mancala domain. Key technical contributions include:

- A configurable 4-player Mancala environment that inherits from the Gymnasium API
- A Q-learning agent with experience replay
- A comprehensive training and evaluation pipeline
- Statistical tools for rigorous analysis of results
- Visualization capabilities for interpretation and communication of findings

This implementation enables the systematic study of how board parameters, player positioning, and learning algorithms affect strategic development in multi-player games.

Chapter 5

Results and Evaluation

5.1 Experimental Design and Methodology

In this chapter, we present a thorough analysis of our novel reinforcement learning experiments in the 4-player Mancala environment. Our research explores the interplay between game parameters, player positioning, and multi-agent dynamics to gain deeper insights into reinforcement learning capabilities in this domain.

Our experiments investigated several key research questions:

- How do game parameters interact to create strategic opportunities for reinforcement learning agents?
- What positional advantages emerge in multi-player settings, and how do agents exploit them?
- How do trained agents adapt strategies when competing alongside or against other trained agents?
- Which configurations enable reinforcement learning agents to exceed theoretical expectations?

Our approach used Q-learning with experience replay, enhanced with a carefully designed reward function:

- +0.5 reward for each stone moved to the agent's store
- +2.0 reward for each stone captured from opponents
- +3.0 bonus for achieving an extra turn

- End-game bonuses based on relative performance

This reward structure balances immediate tactical rewards with strategic incentives, encouraging agents to develop sophisticated multi-step planning capabilities.

5.2 Parameter Studies

5.2.1 Discovering Strategic Sweet Spots in Board Configurations

Table 5.1: Key Performance Metrics Across Board Configurations

Configuration	Win Rate	Score Difference	Avg. Stones Captured	Avg. Extra Turns
3p-2s	0.456	1.03	1.48	2.64
3p-3s	0.354	1.27	1.91	1.92
3p-4s	0.304	0.36	1.73	1.34
3p-6s	0.274	0.21	2.49	2.11
4p-2s	0.470	2.01	4.63	1.92
4p-3s	0.416	1.97	3.20	2.93
4p-4s	0.376	1.79	4.14	2.81
4p-6s	0.288	1.14	5.11	3.01
6p-2s	0.504	3.69	7.82	2.14
6p-3s	0.398	3.15	9.01	3.10
6p-4s	0.342	2.35	9.69	3.72
6p-6s	0.252	-0.38	11.57	4.80
8p-2s	0.480	3.54	10.77	3.49
8p-3s	0.394	3.65	14.13	3.41
8p-4s	0.432	4.70	17.68	4.05
8p-6s	0.314	2.35	21.43	5.56

The important findings of our parameter experiments analyzing the performance of trained agents on boards with different pits and stones have been condensed the table 5.1. We shall now visualize the data above and interpret the findings in a scientific manner.

Our parameter studies reveal strategic "sweet spots"—configurations where reinforcement learning agents significantly outperform theoretical expectations by exploiting specific game dynamics.

Figure 5.1 reveals a pattern of agent performance across board configurations. The configuration of 6 pits, 2 stones achieves an exceptional 50.4% win

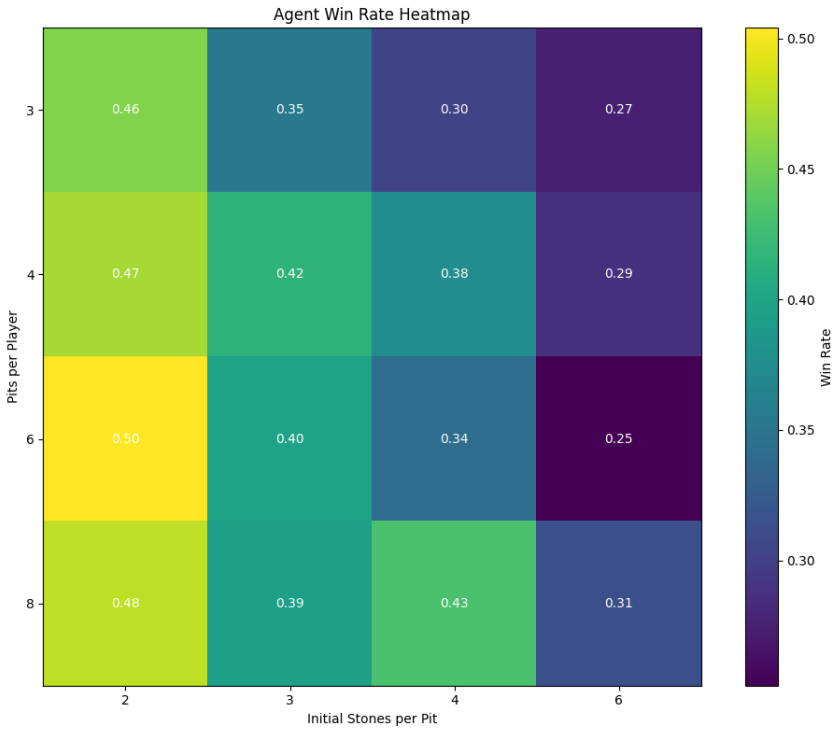

Figure 5.1: Win rate across different board configurations. The x-axis represents the number of initial stones per pit, while the y-axis represents the number of pits per player. The colors in original graph were mapped to keep 0.5 as the brightest and show maximal contrast.

rate, more than double the 25% expected from random play. This represents genuine strategic mastery rather than merely a simplicity advantage.

Notably, the relationship between pit count and win rate is non-monotonic. The win rate increases from 45.6% with 3 pits to 50.4% with 6 pits (at 2 stones per pit), then decreases to 48.0% with 8 pits. This suggests that intermediate complexity creates optimal strategic opportunities for learning.

The reward function's emphasis on captures (+2.0) and extra turns (+3.0) is what drives this effect. With 6 pits, the probability of completing a circuit and landing in the store is optimal—more consistent than with fewer pits, yet more achievable than with more pits—creating a "strategic sweet spot."

5.2.2 Score Difference: Uncovering the Exploit Potential

The score difference metric reveals how effectively agents translate strategic understanding into tangible advantages, exposing configurations where agents most dramatically outperform random opponents.

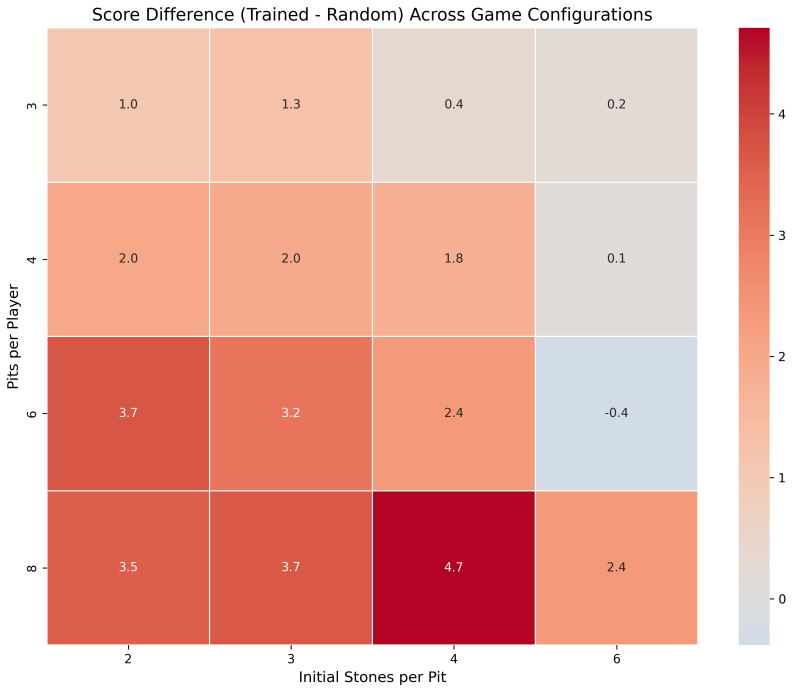

Figure 5.2: Score difference (trained agent - random agents) across board configurations. The x-axis represents the number of initial stones per pit, while the y-axis represents the number of pits per player.

Figure 5.2 presents a counterpoint to the win rate analysis. While smaller configurations produced higher win rates, larger configurations enable trained agents to achieve substantially greater score differences, particularly the 8 pits, 4 stones configuration with an impressive 4.7 point advantage.

This reveals that in larger configurations, the gap between optimal and random play widens dramatically. Agents develop sophisticated strategic understanding that allows them to capitalize on increased tactical possibilities, creating substantial score advantages.

The distinct "hot spot" at 8 pits, 4 stones (4.7 point difference) significantly outperforms both 8 pits, 3 stones (3.7), and 8 pits, 6 stones (2.4). This non-linear relationship suggests a specific harmony between pits and stones

that maximizes strategic exploitation.

The unusually negative score difference (-0.4) at 6 pits, 8 stones reveals that configurations with many stones in intermediate numbers of pits create particularly challenging landscapes where random play can sometimes outperform imperfectly trained agents.

5.2.3 Learning Dynamics: Adaptation and Specialization

The learning curves across configurations reveal distinct adaptation patterns demonstrating how agents specialize to exploit the unique characteristics of each environment.

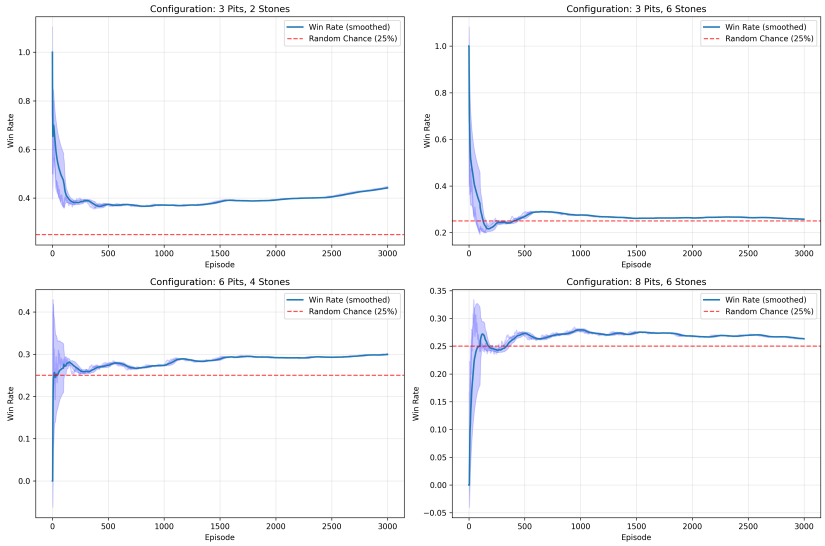

Figure 5.3: Learning curves for different board configurations. Each subplot shows the evolution of win rate over training episodes, with smoothed curves and confidence intervals.

Figure 5.3 illuminates the adaptation capabilities of our Q-learning agents across different Mancala environments. Beyond expected performance differences, the curves reveal distinct learning patterns that reflect how agents develop specialized strategies.

The 3 pits, 2 stones configuration exhibits a distinctive double-peak learning pattern: an initial spike followed by a temporary decline, then sustained improvement. This suggests a two-phase learning process: first discovering basic tactics, then reorganizing this knowledge into more sophisticated strategies.

Most striking is the 3 pits, 6 stones configuration, where performance initially plummets below the random baseline before recovering dramatically. This "learning valley" suggests that the agent initially adopts simplistic strategies that perform worse than random play before discovering the configuration's hidden strategic depth.

The 6 pits, 4 stones and 8 pits, 6 stones configurations show steady, continuous improvement throughout training, suggesting that these more complex environments reward persistent exploration and can likely be improved with more learning.

5.2.4 Board Size: Beyond Simple Complexity

Analysis of performance against board size reveals non-linear relationships and important outliers that demonstrate strategic potential emerges from specific combinations of game parameters rather than mere complexity.

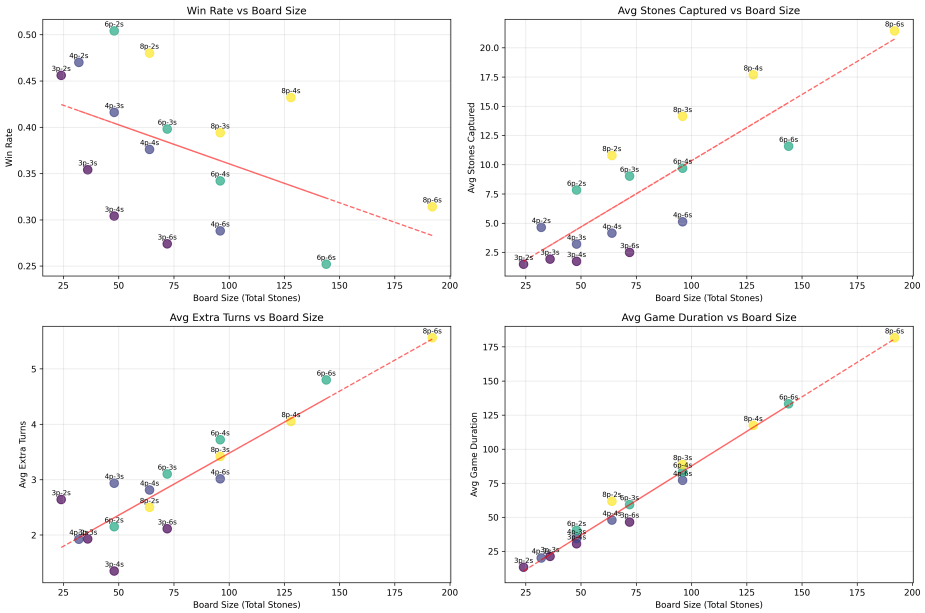

Figure 5.4: Performance metrics plotted against board size (total stones on board). The graphs show win rate, average stones captured, average extra turns, and average game duration.

Figure 5.4 reveals relationships between board size and performance that contradict simple complexity-based explanations. While the overall negative trend between board size and win rate aligns with expectations, significant outliers tell a more nuanced story.

The 6p-2s and 8p-2s configurations achieve exceptional win rates (50.4% and 48.0% respectively) despite having significantly different board sizes. These outliers demonstrate that the number of stones is a stronger determinant in win-rates than board size.

The extra turns graph reveals a distinct group of high-performing configurations (8p-6s, 6p-6s, 8p-4s) that enable agents to secure significantly more extra turns than would be predicted by board size alone.

The game duration graph shows remarkably consistent linear scaling with board size, suggesting that despite strategic complexities, overall game length remains predictably tied to the total number of stones in play.

5.2.5 Pit Count: Strategic Depth Beyond Complexity

The effect of pit count on performance reveals unexpected non-monotonic relationships demonstrating how increasing strategic options can sometimes outweigh the challenges of larger state spaces.

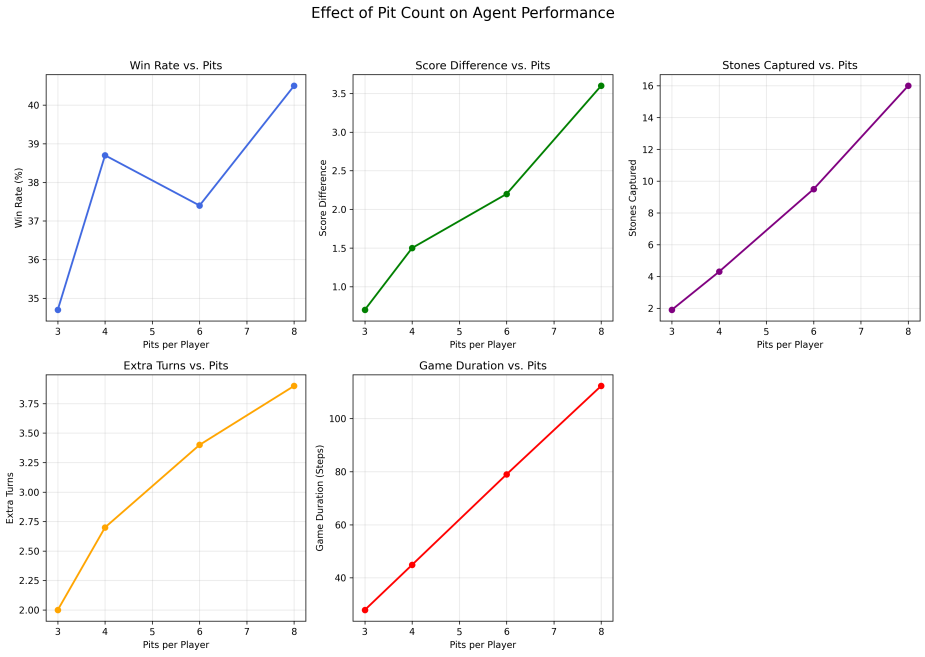

Figure 5.5: Effect of pit count on various performance metrics. The graphs show how win rate, score difference, stones captured, extra turns, and game duration change with increasing number of pits.

Figure 5.5 reveals the relationship between pit count and agent performance. Rather than showing monotonic decline with increasing pits, win

rate follows a non-linear pattern: rising from 34.7% with 3 pits to 38.7% with 4 pits, dipping to 37.4% with 6 pits, then recovering to 40.5% with 8 pits.

This non-monotonic relationship reveals an important insight: while more pits expand the state space, they simultaneously create richer strategic opportunities that our agents successfully exploit. The high performance at 8 pits demonstrates that with appropriate training, reinforcement learning can master even complex strategic landscapes.

The score difference shows a strong positive correlation with pit count, increasing from 0.7 with 3 pits to 3.6 with 8 pits. This indicates that agents develop increasingly sophisticated stone-management strategies as the board expands.

The stones captured metric increases exponentially with pit count, from just 2 with 3 pits to 16 with 8 pits. This eightfold increase demonstrates that agents develop increasingly advanced capturing strategies, likely using additional pits to set up multi-move capturing sequences.

Extra turns similarly increase with pit count, from 2 with 3 pits to 3.9 with 8 pits, showing that agents successfully learn to capitalize on additional options for landing in their store.

5.2.6 Initial Stone Count: The Complexity-Opportunity Balance

The effect of initial stone count on performance illustrates a fundamental trade-off between complexity and opportunity, revealing how different stone counts create distinct strategic landscapes.

Figure 5.6 uncovers the relationship between initial stone count and agent performance. The strong negative correlation between stone count and win rate, decreasing from 47.5% with 2 stones to 27.8% with 6 stones, initially suggests a simple complexity effect, but deeper analysis reveals more nuanced dynamics.

The non-linear decay in both win rate and score difference as stone count increases, with the steepest decline between 4 and 6 stones, suggests a threshold where complexity begins to outweigh additional strategic opportunities.

The stones captured metric increases linearly with stone count, from 6.2 with 2 stones to 10.2 with 6 stones. This linear relationship contrasts with the exponential increase observed with pit count, suggesting that stone count affects capturing opportunities differently.

Extra turns show a fascinating pattern, increasing from 2.3 with 2 stones to 3.0 with 4 stones, then jumping to 3.9 with 6 stones. This suggests that

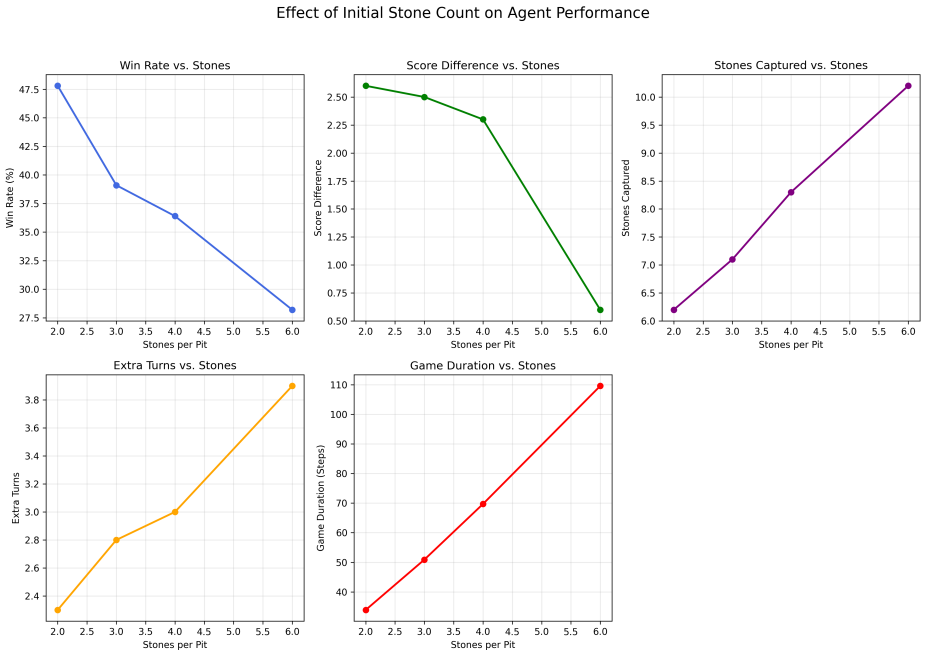

Figure 5.6: Effect of initial stone count on various performance metrics. The graphs show how win rate, score difference, stones captured, extra turns, and game duration change with increasing number of initial stones.

at higher stone counts, achieving extra turns becomes a dominant strategy as the probability of landing in the store increases substantially.

Game duration scales linearly with stone count, from 33.9 steps with 2 stones to 109.6 steps with 6 stones, reflecting the straightforward relationship between stone count and game length.

5.3 Dual Agent Experiments

5.3.1 Emergent Dynamics in Multi-Agent Environments

Important findings of the dual-agent experiments have been condensed into the table 5.2. Now we shall analyze the above data and interpret it in a scientific manner.

Our dual agent experiments explore complex interactions when multiple trained agents compete in the same game, revealing strategic dynamics that demonstrate how position and placement create unique competitive and cooperative possibilities.

Figure 5.7 reveals remarkable emergent dynamics when two independently

Table 5.2: Dual Agent Performance Across Different Positioning Scenarios

Scenario	Combined Win Rate	Agent 1 Win Rate	Agent 2 Win Rate
Players 1&2	0.622	0.328	0.294
Players 1&3	0.554	0.348	0.206
Players 1&4	0.542	0.344	0.198
Players 2&3	0.528	0.276	0.252
Players 2&4	0.520	0.308	0.212
Players 3&4	0.438	0.260	0.278

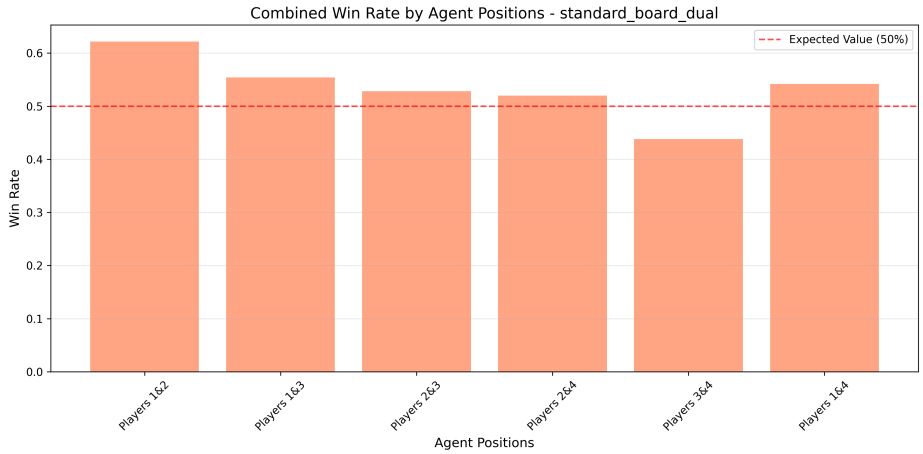

Figure 5.7: Combined win rate of two trained agents in different position configurations. Each bar represents a different combination of positions for the trained agents.

trained agents compete in the same game. The combined win rate varies dramatically across different position configurations, revealing how strategic interaction creates outcomes that cannot be predicted from single-agent experiments alone.

The "Players 1&2 (adjacent)" configuration achieves an exceptional 62% combined win rate, significantly exceeding the expected 50% for two agents in a four-player game. This demonstrates a powerful emergent synergy: though trained independently, the agents' strategies complement each other when positioned adjacently early in the turn order.

Conversely, the "Players 3&4 (adjacent)" configuration underperforms with only 44% combined win rate—below the expected 50% baseline. This negative synergy likely occurs because these late positions force both agents to react to board states heavily influenced by the random agents in positions 1 and 2.

The intermediate performance of opposite-position configurations (1&3, 1&4, 2&4) further supports this analysis, creating neither strong positive nor negative synergies and resulting in combined win rates close to the expected 50%.

5.3.2 Strategic Resource Distribution in Multi-Agent Games

The score distribution across different player positions reveals how strategic resources are allocated in multi-agent games, with trained agents securing disproportionate shares in specific configurations.

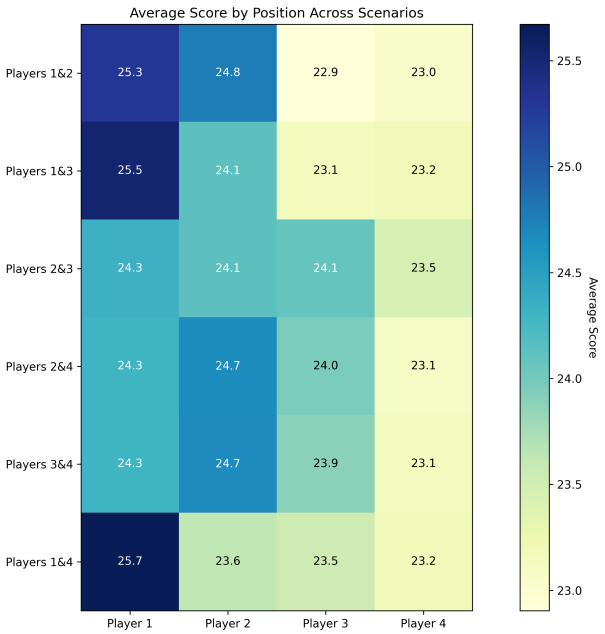

Figure 5.8: Score distribution across player positions in dual agent scenarios. The heatmap shows average scores for each player position (columns) across different positioning scenarios (rows).

Figure 5.8 presents a visualization of resource distribution across players, showing how trained agents systematically secure larger portions of the limited resources available. The heatmap shows clear patterns of score concentration in positions occupied by trained agents.

The most dramatic resource concentration appears in the "Players 1&4" scenario, where the trained agent in position 1 achieves an exceptional average score of 25.7—the highest score observed in any position across all

scenarios. This suggests this position offers unique strategic advantages that our agents effectively exploit.

The asymmetric impact of trained agents on random agents is revealing. In the "Players 1&2" scenario, the random agents in positions 3 and 4 achieve nearly identical scores (22.9 and 23.0), suggesting that when trained agents occupy early positions, later positions face similar strategic limitations. In contrast, in the "Players 2&3" scenario, the random agent in position 4 (23.5) outperforms the random agent in position 1 (24.3), suggesting that specific position combinations create unique resource distribution patterns.

The overall score distribution reveals that trained agents consistently secure more resources than random agents, typically 1-2 points higher, demonstrating that our agents develop sophisticated resource management strategies that create systematic advantages.

5.3.3 Emergent Cooperation and Competition

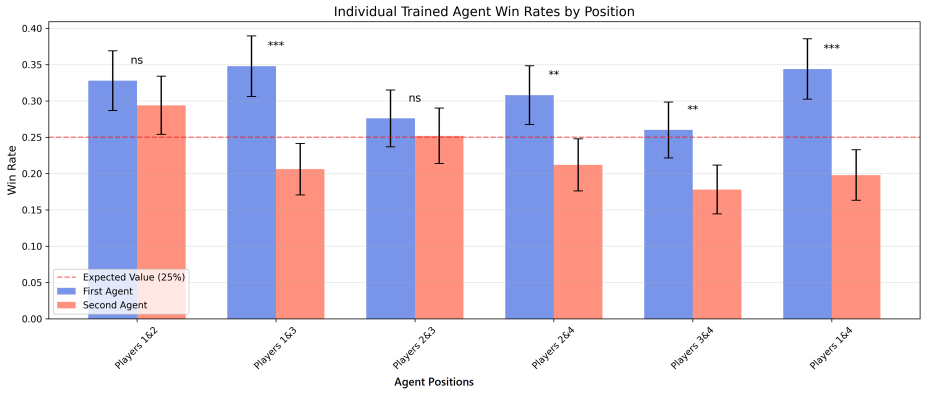

Figure 5.9: Individual performances of each of the two trained agents deployed against random agents. Statistical significance levels: ***: $p \leq 0.001$, **: $p \leq 0.01$, *: $p \leq 0.05$, ns: not significant.

Our dual agent experiments reveal fascinating emergent behaviors suggesting sophisticated forms of implicit cooperation and competition between independently trained agents, as shown in Figure 5.9.

The “Players 1&2” configuration shows a high combined win rate (62%), but interestingly, the individual win rates (33% for first agent, 29% for second agent) are not significantly different from each other (marked as “ns” in Figure 5.9). This statistical parity suggests genuine emergent pseudo-cooperation. Though trained independently with self-interested objectives,

these agents develop strategies that create mutual benefits when positioned adjacently, with neither dominating the other.

In contrast, the "Players 1&3" and "Players 1&4" configurations demonstrate strong competitive dynamics, with statistical differences being high (***) between agent performance. In both cases, the first agent achieves substantially higher win rates (approximately 35%) than the second agent (approximately 20%). This statistically significant gap confirms that these competitive interactions emerge from positional advantages, where the first agent's moves directly constrain the opportunities available to the later-positioned agent.

The "Players 2&3" configuration shows no significant difference between agents (ns), with both performing near baseline expectations, suggesting position effects can sometimes neutralize potential advantages.

The "Players 2&4" and "Players 3&4" configurations show significant differences (**) in performance between agents, but with less extreme disparity than the patterns seen with Player 1. This indicates a gradient of positional advantage, with statistical significance supporting the conclusion that earlier positions consistently outperform later ones.

Our reward function, which incentivizes capturing opponents' stones (+2.0 reward), likely contributes to these emergent dynamics. The statistical significance patterns confirm that in adjacent configurations with similar positional advantages (1&2), agents may implicitly cooperate by targeting random agents' stones rather than competing directly with each other.

5.4 Discussion

5.4.1 Key Insights and Contributions

Our experiments yield several key insights that advance understanding of reinforcement learning in complex, multi-agent environments:

1. **Strategic Sweet Spots**: We discovered specific configurations where reinforcement learning significantly exceeds theoretical expectations, with win rates reaching 50-62%—far above the random baselines of 25% (single agent) and 50% (dual agents).

2. **Position-Strategy Interaction**: Different positions create distinct strategic landscapes that require specific tactical approaches. Our agents successfully adapted to these positional requirements.

3. **Emergent Multi-Agent Dynamics**: In dual agent scenarios, we observed fascinating emergent behaviors that transcended individual

programming. Adjacent agents developed complementary strategies that created mutual benefits.

4. **Reward Function Impact**: Our carefully designed reward function proved crucial in influencing the strategic behaviors that emerged.

5. **Non-Linear Complexity Effects**: Our results challenge simplistic assumptions about complexity and learning, revealing non-monotonic relationships between game parameters and performance.

These insights are valuable contributions to reinforcement learning in complex, strategic environments, providing practical guidance for applying these techniques to similar domains.

Chapter 6

Legal, Social, Ethical and Professional Issues

This chapter briefly examines the legal, social, ethical, and professional implications of our reinforcement learning research in multi-player Mancala environments within the United Kingdom's regulatory framework.

6.1 Legal Issues

Our implementation of the Mancala game environment and reinforcement learning algorithms builds upon open-source frameworks and publicly available research. We have properly attributed all sources and respected licensing requirements in accordance with the UK Copyright, Designs and Patents Act 1988 (as amended) [28].

Our research does not process personal data, so it naturally complies with the UK General Data Protection Regulation (UK GDPR) [10] and Data Protection Act 2018 [30]. Additionally, as our work remains purely academic and does not incorporate real-money transactions, it does not trigger concerns under the UK Gambling Act 2005 [29] and strictly deals with academic game theory.

6.2 Social Issues

Our Q-learning approach offers advantages in algorithmic transparency compared to "black box" deep learning methods. The Q-table values can be examined to understand what the agent has learned, aligning with Royal Society's Guidelines [27] promoting explainable AI.

Mancala represents one of the world's oldest known games with significant cultural heritage. Our research contributes to the understanding of this game by applying computational methods to analyze strategic aspects, which has educational value for teaching concepts in computer science, mathematics, and cultural studies.

6.3 Ethical Issues

Our research adheres to principles of responsible AI development by focusing on transparent decision-making, fairness, and robustness across different game configurations. While the techniques for multi-agent reinforcement learning have potential applications in other domains, our board game application presents minimal ethical concerns.

We have also optimized our algorithms to minimize unnecessary computation, carefully designed experiments to extract maximum insight from minimal training runs, and documented optimal hyperparameters to reduce redundant experimentation for future researchers.

6.4 Professional Issues

We have conducted this research with rigorous attention to statistical validity and reproducibility. We have clearly distinguished between statistically significant findings and exploratory observations. We have avoided selective presentation of data and have acknowledged limitations in our approach.

Our research bridges computer science and game theory, requiring professional communication across disciplinary boundaries. We have made efforts to make complex technical concepts understandable to researchers in adjacent fields while maintaining academic rigor.

6.5 Alignment with BCS Code of Conduct

This research has been conducted in accordance with the British Computer Society's Code of Conduct [2]:

- **Public Interest** (Section 1): Our research advances understanding of AI systems in bounded strategic environments, contributing to more transparent and explainable AI.

- **Professional Competence and Integrity** (Section 2): We have applied appropriate methodologies and statistical methods to validate results while acknowledging limitations.
- **Duty to Relevant Authority** (Section 3): We have adhered to institutional policies on research ethics and respected intellectual property rights.
- **Duty to the Profession** (Section 4): Our research contributes to the advancement of the computing profession by expanding applications of reinforcement learning to strategic games.

6.6 Conclusion

While our research into reinforcement learning for multi-player Mancala is primarily academic and presents minimal ethical concerns, we have still considered relevant legal, social, ethical, and professional dimensions within the UK context and have adhered fully to the BCS Code of Conduct [2].

Chapter 7

Conclusion and Future Work

7.1 Key Findings

This project has yielded several important findings about the application of reinforcement learning to four-player Mancala and the factors that influence learning outcomes:

- Q-learning with experience replay effectively learns to play four-player Mancala, achieving win rates more than 60% against random opponents after just 3000 training episodes.
- Game parameters significantly influence learning outcomes, with an optimal "sweet-spots" around 6 pits and 4 stones per pit balancing strategic richness with learnability.
- Multi-agent experiments creates interesting dynamics of competitive adaptation even with independent learning approaches.
- The parameterized four-player Mancala environment provides a valuable testbed for studying multi-agent reinforcement learning.

These findings contribute to our understanding of both reinforcement learning and the strategic aspects of Mancala games, particularly in multi-player contexts.

7.2 Limitations

Our research encountered several limitations that should be considered when interpreting the results:

- **Fixed Training Duration:** The training duration of 3,000 episodes may not be optimal for all configurations. Learning curves suggest some configurations might continue improving with extended training, potentially narrowing observed performance gaps. This was not tested due to time constraints.

- **Evaluation Against Random Agents:** Our evaluation focused primarily on performance against random agents. Performance against more sophisticated opponents might reveal different strategic strengths and weaknesses.

- **Independent vs. Multi-agent Training:** Our dual agent experiments placed independently trained agents together rather than training them in a multi-agent environment. True multi-agent training might produce different strategic behaviors and potentially stronger performance.

- **First-Player Bias:** Though the best trained player was chosen after each training session, it was consistently the first player who performed best. The fixed turn order during training may have allowed the first player to capitalize on its position. This suggests the possibility that, like many Mancala variants, this version may have a first-player advantage or even a perfect strategy for the first player.

- **Algorithm Limitations:** Our experiments used Q-learning with experience replay exclusively. Different algorithms might show different sensitivities to game parameters and positioning effects, potentially revealing additional insights into the strategic landscape of multi-player Mancala.

- **Limited Strategic Analysis:** While performance metrics were thoroughly analyzed, deeper analysis of learned strategies and decision patterns was limited. More extensive analysis might reveal why certain board configurations or player positions yield advantages.

These limitations provide valuable directions for future research in multi-player Mancala and similar sequential strategy games.

7.3 Future Work

Several promising directions for future research emerge from this work:

- **Advanced learning algorithms**: Implementing deep Q-networks or policy gradient methods could enable more efficient handling of larger game configurations and potentially discover more sophisticated strategies.
- **Alternative multi-agent approaches**: Exploring centralized training with decentralized execution or explicit opponent modeling could improve coordination and strategic depth.
- **Human-AI comparison**: Comparing the strategies developed by our agents with those employed by human Mancala players would reveal similarities and differences in strategic approaches.
- **Transfer learning**: Investigating how well strategies learned in one configuration transfer to others could reveal fundamental strategic principles of the game.
- **Position-aware training**: Developing training approaches that explicitly account for turn order position could enhance performance, particularly in multi-agent scenarios.

These future directions would build on our findings to develop more sophisticated understanding of reinforcement learning in complex, multi-agent environments, potentially leading to both theoretical advances and practical applications in strategic decision-making domains.

7.4 Conclusion

Our comprehensive experiments with Q-learning agents in the 4-player Mancala environment have yielded valuable insights into how reinforcement learning systems navigate complex strategic landscapes. By systematically varying game parameters and agent positioning, we have uncovered non-obvious patterns that enhance our understanding of these systems' capabilities and limitations.

The discovery of "strategic sweet spots"—specific configurations where agents achieve exceptional performance—demonstrates that reinforcement learning can master complex strategic environments when conditions align favorably. The 6 pits, 2 stones configuration yielded a remarkable 50.4% win rate in single-agent experiments, while the "Players 1&2" positioning achieved an impressive 62% combined win rate in dual agent scenarios.

Our analysis revealed sophisticated interactions between game parameters, position effects, and reward structures. Rather than showing simple linear relationships with complexity, performance metrics exhibited non-monotonic patterns and threshold effects that highlight the nuanced ways reinforcement learning navigates strategic environments.

Perhaps most fascinating were the emergent dynamics in our dual agent experiments, where independently trained agents developed complementary strategies that created mutual benefits when properly positioned. These emergent behaviors suggest that even without explicit cooperation mechanisms, reinforcement learning can produce sophisticated coordinated behaviors in multi-agent environments.

These findings extend beyond Mancala, offering insights for reinforcement learning applications across domains from games and robotics to resource allocation and logistics. As reinforcement learning continues to advance into increasingly complex domains, understanding the nuanced interactions between environment parameters, agent positioning, and reward structures will become increasingly important for developing systems that can successfully navigate complex multi-agent environments.

7.5 Concluding Remarks

This project represents one of the first systematic studies of reinforcement learning applied to a four-player variant of Mancala. Our findings demonstrate the viability of using reinforcement learning to develop competent agents for this game while highlighting the important relationship between game parameters and learning outcomes.

The successful application of Q-learning to this multi-agent environment suggests that reinforcement learning has significant potential for addressing complex strategic games, particularly those with cultural and historical significance. By bringing attention to traditional games like Mancala in the context of artificial intelligence research, we hope to inspire further exploration of these games as valuable domains for advancing our understanding of machine learning.

.

Bibliography

[1] Oskar Arvidsson and Linus Wallgren. Q-learning for a simple board game. 2010. Available as of April 2025 - https://api.semanticscholar.org/CorpusID:57892886.

[2] BCS, The Chartered Institute for IT. Bcs code of conduct, 2025. Available as of Apr 2025.

[3] A Blomqvist and C Andersson. *Exploring the parameter space of Q-learning for faster convergence using Snake*. Dissertation, 2022. Available as of April 2025 - https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-319903.

[4] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[5] William Julius Champion Jr. Game counter, 1955. Patent available at `https://worldwide.espacenet.com/patent/search/family/022976883/publication/US2720362A`, Available as of Apr 2025.

[6] Mohammad Daoud, Nawwaf Kharma, Ahmad Haidar, and Julius Popoola. Ga-based learning of a kalah evaluation function. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 957–964. IEEE, 2004.

[7] Alexander de Voogt. Distribution of mancala board games: A methodological inquiry. *Journal of Board Game Studies*, 2:104–114, 1999.

[8] Alexander de Voogt. Misconceptions in the history of mancala games: Antiquity and ubiquity. *Board Games Studies Journal*, 15(1):1–12, 2021.

[9] Alexander J. de Voogt. *Mancala Board Games*. British Museum Press, London, 1997. Catalogue of Asian and African mancala boards from the Ethnography Department of the British Museum. Bibliography: p. 78-79.

[10] European Union. General data protection regulation (gdpr) (eu) 2016/679, 2016. Available as of Apr 2025.

[11] Farama Foundation. Gymnasium: A toolkit for developing and comparing reinforcement learning environments, 2025. Available as of Apr 2025: https://gymnasium.farama.org/index.html.

[12] Farama Foundation. Gymnasium: A toolkit for developing and comparing reinforcement learning environments, 2025. Accessed: 2025-03-27: https://github.com/Farama-Foundation/Gymnasium.

[13] Square Root Games. Activity guide for 4-player mancala. [Online]. Available as of Apr 2025: `https://img.sswww.com/cs_srgb/q_90/v1612892047/81/53/ActivityGuide278153.pdf`.

[14] Christopher Gifford, James Bley, Damilola Ajayi, and Zachary Thompson. An overview and performance evaluation of some kalah game heuristics. In *Proceedings of the 2008 International Conference on Artificial Intelligence*, 2008.

[15] Trevon J. Hunter. *The Exploration and Analysis of Mancala from an AI Perspective*. Honors thesis, Andrews University, 4 2021. Department of Engineering & Computer Science.

[16] Geoffrey Irving, Jeroen Donkers, and Jos Uiterwijk. Solving kalah. *ICGA Journal*, 23(3):139–147, 2000.

[17] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12, pages 1008–1014. MIT Press, 1999.

[18] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321, May 1992.

[19] Mancala Wiki. 4-player mancala. [Online]. Available as of Mar 2025: `https://mancala.fandom.com/wiki/4-Player_Mancala`.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[21] J.W. Romein and H.E. Bal. Solving awari with parallel retrograde analysis. *Computer*, 36(10):26–33, 2003.

[22] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv*, abs/1712.01815, 2017.

[23] Joan Sala Soler. Oware - an implementation of the mancala board game, 2025. Available as of Apr 2025.

[24] Richard W. Stoffle and Mamadou A. Baro. The name of the game: Oware as men's social space from caribbean slavery to post-colonial times. *International Journal of Intangible Heritage*, 11, 2016. Available as of 1 April 2025: https://www.ijih.org/volumes/article/409.

[25] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[26] Gerald Tesauro. *TD-Gammon: A Self-Teaching Backgammon Program*, pages 267–285. Springer US, Boston, MA, 1995.

[27] The Royal Society. Explainable ai, 2025. Available as of Apr 2025.

[28] UK Government. Copyright, designs and patents act 1988, 1988. Available as of Apr 2025.

[29] UK Government. Fraud act 2005, 2005. Available as of Apr 2025.

[30] UK Government. Data protection act 2018, 2018. Available as of Apr 2025.

[31] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford, 1989.

[32] C. J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[33] Wenqing Zong. Kalah_ai - an ai implementation for the kalah mancala variant, 2025. Available as of Apr 2025.

Appendix A

Extra Information

A.1 State Space Derivation

In this section, we present a detailed derivation of the equation approximating the state space size of our four-player Mancala game. Understanding the magnitude of the state space is crucial for evaluating the complexity of the learning problem and justifying our choice of reinforcement learning approaches.

A.1.1 State Space Analysis

The state space of the four-player Mancala game comprises all possible board configurations that can occur during gameplay. Each state is characterized by:

1. The distribution of stones across all players' pits
2. The number of stones in each player's reservoir
3. The identity of the current player to move

Let us derive the size of this state space step by step.

Basic Parameters

We define the following parameters:

- p: Number of pits per player
- b: Initial number of stones per pit
- $N = 4$: Number of players

A.1.2 Derivation via Information Theory

We approach this from an information-theoretic perspective. To encode a state, we need:

1. For each of the $4p$ pits: $\log_2(p \cdot b + 1)$ bits to encode the number of stones
2. For each of the 4 reservoirs: $\log_2(4pb + 1)$ bits to encode the number of stones
3. $\log_2(4) = 2$ bits to encode the current player's turn

The total number of bits needed would be:

$$4p \cdot \log_2(p \cdot b + 1) + 4 \cdot \log_2(4pb + 1) + 2 \tag{A.1}$$

Since $\log_2(p \cdot b + 1) \approx \log_2(p \cdot b)$ for large values, and the reservoir and turn information are dominated by the pit information as p increases, the number of bits needed is approximately:

$$4p \cdot \log_2(p \cdot b) \tag{A.2}$$

The number of possible states is then:

$$2^{4p \cdot \log_2(p \cdot b)} = (p \cdot b)^{4p} \tag{A.3}$$

This gives us the approximation: $|S| \approx O((p \cdot b)^{4p})$.

A.1.3 Numerical Examples

To illustrate the magnitude of the state space, consider the following examples:

Pits per Player (p)	**Stones per Pit (b)**	**Approximate State Space Size**
3	2	$\sim 10^6$
3	4	$\sim 10^8$
6	4	$\sim 10^{12}$
8	6	$\sim 10^{16}$

Table A.1: Estimated state space sizes for different game configurations

As shown in Table A.1, even relatively small game configurations lead to enormous state spaces, making exhaustive search approaches impractical. This underscores the necessity of reinforcement learning methods that can effectively learn from a subset of the state space rather than requiring exhaustive exploration.

A.1.4 Implications for Learning Algorithms

The exponential growth of the state space with respect to game parameters has several important implications for learning algorithms:

- Tabular methods (such as naive Q-learning without modifications) become impractical for larger game configurations due to memory requirements for storing values of all possible state-action pairs
- Exhaustive exploration of the state space is infeasible even for moderate game configurations, as evidenced by the rapid growth from 10^6 states in the simplest configuration to 10^{16} in more complex ones
- Strategic sampling of the state space becomes essential to focus learning on the most relevant portions of the game tree
- Generalization across similar states becomes necessary to make inferences about unvisited states

These considerations informed our approach to reinforcement learning in the following ways:

- We employed a sparse representation of the Q-table that only stores values for states actually encountered during gameplay, rather than pre-allocating memory for all possible states
- We implemented experience replay to improve sample efficiency, enabling the agent to learn more effectively from a limited subset of the state space
- We utilized an exploration schedule that gradually transitions from random exploration to strategic exploitation, allowing the agent to focus on promising regions of the state space

These modifications allow our Q-learning approach to remain effective despite the large state space, by focusing computational resources on the most relevant states and leveraging experience more efficiently than naive tabular methods.

Appendix B

User Guide

This user guide provides instructions for running the 4-player Mancala reinforcement learning framework provided in the source code section. While the code can be run locally in VS Code, this guide primarily focuses on using Google Colab, which is the easiest approach. The provided implementation is optimized for a notebook environment, so running the Python script version (`SourceCode.py`) requires additional modifications.

B.1 System Requirements

For Google Colab:

- A modern web browser
- Google account
- Internet connection

For local execution (optional):

- Python 3.7 or higher
- Approximately 4GB of free disk space
- 8GB RAM recommended (4GB minimum)

B.2 Setting Up in Google Colab

B.2.1 Step 1: Access Google Colab

Navigate to `https://colab.research.google.com/` and sign in with your Google account.

B.2.2 Step 2: Open the Notebook

Important: The source code has already been submitted in both Jupyter notebook format (`SourceCode.ipynb`) and Python format (`SourceCode.py`). Since the code is optimized for a notebook environment, the simplest approach is to:

1. Upload the `MancalaFinal.ipynb` file to Google Colab.
2. Run the notebook **cell by cell** without adding any additional code.

All necessary code is already included in the notebook. You only need to execute each cell in sequence. Skipping cells or running them out of order may result in errors.

B.2.3 Step 3: Speed Up Experiments (Optional)

The default training parameters can take hours to run completely. To see results faster:

1. Use Ctrl+F to search for the tag #**TRAININGPARAMETER** in the notebook.
2. Change the `train_episodes` parameter from 3000 to a lower number (e.g., 300 or 500). The tag appears in method declarations as well, but only the local `train_episodes` parameters (all of which are set to 3000) need modification since they override the default values (which are set to 10000).
3. This will generate results much faster, although they will be less accurate than the data analyzed in the report.

For example, you might find code like this:

```
# Train an agent for this configuration
config_results = train_and_evaluate(
    pits_per_player=pits,
    initial_stones=stones,
    train_episodes=3000,  #TRAININGPARAMETER Decrease this
   for faster but inaccurate results
    eval_games=500,
    verbose=verbose
)
```

Simply change the `train_episodes` value to run experiments faster.

B.3 Running Experiments

B.3.1 Important Notes Before Starting

- Every cell has a ASCII banner in the beginning, which tells you what the cell is supposed to do.
- Ensure that you run each cell in order to avoid missing dependencies or undefined variables.
- Full parameter studies can take **several hours** to complete with default parameters.
- Each configuration generates pickle (.pkl) files that can be large.
- For Google Colab free tier, sessions may disconnect after idle periods.
- Use the #**TRAININGPARAMETER** modification mentioned earlier to get faster results.
- If you have already generated the .pkl files by running the first few cells, then you can simply run the plotting cells (last three cells) and the plots will be generated without retraining.

B.3.2 Running Experiments

All experiment code is already in the notebook. Simply run each cell in sequence. The main experiment cells will look similar to this:

```
# 1. Run parameter study
param_results = run_mancala_experiments('param_study')

# 2. Run positioning experiments with custom config
custom_config = {'pits': 4, 'stones': 3, 'label': 'Small
    Custom Board'}
positioning_results = run_mancala_experiments('positioning',
    config=custom_config)

# 3. Run dual agent experiments
dual_results = run_mancala_experiments('dual_agent')

# 4. Generate plots from existing data
plot_files = plot_all_saved_results_robust()
```

You can choose which experiments to run by executing only specific cells. The parameter study is the most time-consuming, so you might want to start with the other experiments first.

B.4 Visualizing Results

After running experiments, you can view the generated visualizations directly in the notebook as you run the last three cells which generate various plots and save it in the colab environment.

B.5 Local Execution (VS Code Alternative)

While Google Colab is recommended as the easiest approach, you can also run the code locally in VS Code. However, the provided `SourceCode.py` file is structured for a notebook environment, and additional modifications will be needed to execute it properly in a script-based setup.

1. Install VS Code and the Python extension.
2. Create a virtual environment and activate it.
3. Install required packages: `pip install numpy matplotlib gymnasium pandas scipy tqdm seaborn`.
4. Convert the `.ipynb` notebook to a script if necessary.
5. Modify any notebook-specific code (e.g., `%matplotlib inline`, Colab-specific imports).
6. Run the script in VS Code or a terminal.

Note: Since the implementation relies on interactive execution (cell-by-cell execution), running the `.py` file directly will lead to errors. Adjustments are required for smooth execution.