Design and Evaluation of Mancala AI Agents: A Comparative Study of Heuristic Alpha-Beta Search and Double Deep Q-Networks

Dang Quang Nguyen
Applied Modeling and Quantitative Methods (Big Data Analytics)
Trent University
Peterborough, Canada
dangquangnguyen@trentu.ca

Crystal Wang
Applied Modeling and Quantitative Methods (Big Data Analytics)
Trent University
Peterborough, Canada
crystalwang@trentu.ca

***Abstract*— This paper presents a comparative evaluation of two AI strategies for the strategic board game Mancala (Kalah variant): (1) a heuristic-driven Alpha-Beta pruning agent and (2) a Deep Q-Network (Double DQN) agent enhanced with Prioritized Experience Replay (PER) and curriculum learning. The agents were tested through large-scale simulations against a range of baseline opponents, including Random, Simple Minimax, and standard Alpha-Beta agents. Results show that the heuristic-based Alpha-Beta agent achieves near-perfect performance when playing first, but with significant computational overhead. The Double DQN agent, by contrast, offers a better balance between strategic strength and computational efficiency, achieving an 80.8% win rate against the Random agent after 24,000 training episodes. These findings demonstrate the viability of combining deep reinforcement learning with structured reward engineering in deterministic, turn-based games like Mancala and highlight trade-offs between handcrafted search policies and data-driven approaches.**

Keywords—Mancala, Kalah, heuristic search, Alpha-Beta pruning, Deep Q-Network, reinforcement learning, curriculum learning

I. INTRODUCTION

Mancala, a family of ancient strategy board games played across Africa, the Middle East, and Asia, offers a rich platform for artificial intelligence (AI) research due to its strategic depth, deterministic structure, and complex turn-based dynamics. Among its many variants, **Kalah**, popular in North America, is particularly well-suited for computational modeling. The game challenges players to optimize seed distribution and capture tactics across a fixed set of pits and stores, with rules that encourage long-term planning and adaptive decision-making.

Despite its strategic complexity, Mancala has received comparatively little attention in AI research compared to games like Chess, Go, and Checkers [19]. Most existing Mancala agents rely on traditional search algorithms or handcrafted heuristics, with modern learning-based approaches remaining largely unexplored.

This study addresses this gap by evaluating two high-performing AI agents for the Kalah variant. The first agent combines **Minimax search with Alpha-Beta pruning** and a set of domain-informed heuristic functions to guide evaluation and move ordering. The second leverages a **Double Deep Q-Network (DQN)** enhanced with **Prioritized Experience Replay (PER)** and a **three-phase curriculum learning framework**, enabling it to learn optimal policies from experience.

By benchmarking these agents against a range of baselines, including Random, Simple Minimax, and standard Alpha-Beta, this research investigates the trade-offs between handcrafted strategic reasoning and autonomous learning. The goal is to provide insight into the performance, adaptability, and computational efficiency of traditional search-based versus deep reinforcement learning (DRL) methods in the context of deterministic, turn-based board games like Mancala.

A. *Rules of Kalah*

Kalah is a two-player board game played on a board divided into two symmetrical sides, each featuring six pits and one store. Initially, each pit contains four seeds, as depicted in Fig. 1.

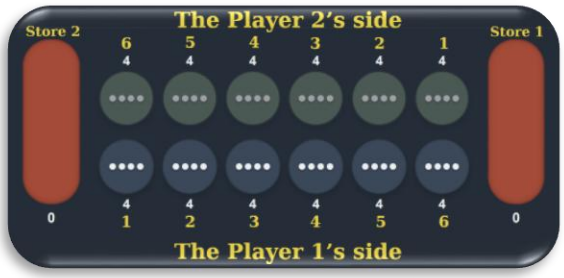


Fig. 1. Initial Kalah board state.

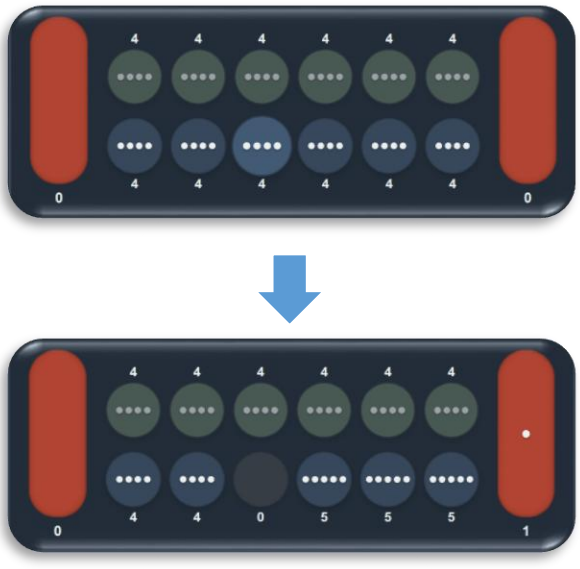
White numbers indicate seed count in each pit. Yellow numbers denote the pit position.

Rule 1: A player selects one of their non-empty pits, collects all seeds from it, and distributes them sequentially into subsequent pits, moving counterclockwise around the board, including the opponent's pits.

Rule 2: During a turn, the player must skip the opponent's store.

Rule 3: If the last seed lands in the player's store, the player earns an additional move (detailed in Fig. 2). For example, Player 1 selects pit 3 from their row, which contains 4 seeds. Thus, the last seed lands in their store, granting them an extra turn.

Before the move

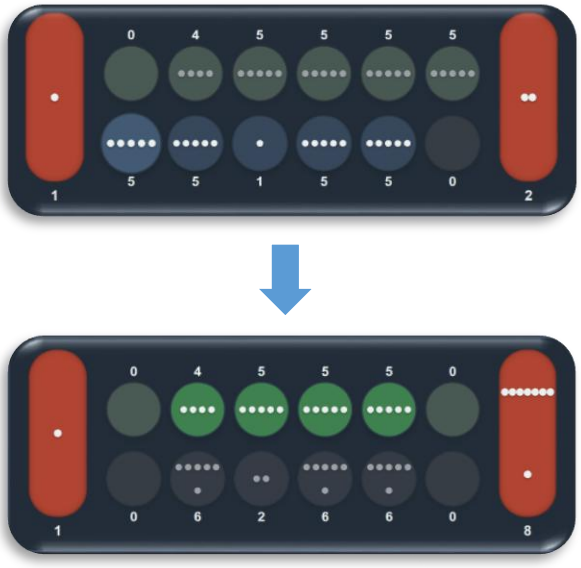


After the move

Fig. 2. Illustration of extra turn condition—last seed sown into player 1's store.

Rule 4: If the final seed is sown in an empty pit on the player's side and the directly opposite pit contains seeds, the player captures all seeds from both pits, transferring them to their store (detailed in Fig. 3). For example, Player 1 selects pit 1 from their row, which contains 5 seeds. Therefore, the last seed ends in pit 6, capturing the opposite pit's seeds of the opponent.

Before the move:



After the move

Fig. 3. Illustration of capture condition—the final seed lands in an empty pit, capturing opponent seeds.

II. Related Work

Artificial intelligence has made substantial advances in strategic board games such as Chess, Go, Checkers, and Othello, where they often serve as benchmarks for search algorithms and deep reinforcement learning (DRL) methods [19]. Meanwhile, the game of Mancala, particularly its Kalah variant, remains significantly underrepresented in peer-reviewed literature. Only a handful of academic studies have focused on developing AI agents for Mancala, with most relying on rule-based or shallow heuristic strategies. For instance, [1] examined heuristic-deterministic models, while [2] implemented classic search algorithms but did not explore learning-based methods. In contrast to well-researched games like Go, which benefit from large-scale datasets, standardized rules, and global competitions, Mancala poses unique challenges such as non-uniform turn order, asymmetric state transitions, and specialized capture mechanics that complicate both search optimization and value function estimation.

These factors make Mancala an ideal but underutilized platform for evaluating general-purpose game-playing agents in deterministic, turn-based environments. This study contributes to filling this gap by introducing and benchmarking two high-performance agents: a hybrid Alpha-Beta pruning agent enhanced with domain-specific heuristics, and a Double Deep Q-Network (DQN) agent augmented with Prioritized Experience Replay (PER) and curriculum learning.

This relative lack of attention makes Mancala a valuable testbed for advancing general-purpose game-playing agents, particularly in deterministic, turn-based environments with sparse rewards and long-term dependencies. By contributing both a hybrid Alpha-Beta heuristic agent and a deep Q-learning framework enhanced with curriculum learning and PER, this study directly addresses the lack of peer-reviewed reinforcement learning applications in Mancala. It offers a structured comparative analysis that contrasts traditional search-based methods with modern learning-based strategies, thereby filling a methodological and empirical void in current literature. Moreover, it expands the diversity of strategic board games explored in reinforcement learning research beyond widely studied domains like Chess and Go.

A. Traditional AI Approaches

[2] investigated several heuristic algorithms integrated with Minimax, Alpha-Beta pruning, and lookahead strategies. Their research involved extensive experimentation through AI-versus-random agent scenarios and round-robin AI tournaments. Metrics assessed included win rate, stones moved per turn, captures per move, and frequency of extra moves. Seven heuristics were examined:

- **H0:** First valid move (furthest from store)
- **H1:** Player's current advantage over the opponent
- **H2:** Proximity to victory (more than half seeds)
- **H3:** Opponent's proximity to victory
- **H4:** Number of seeds near the player's store
- **H5:** Number of seeds distant from the player's store
- **H6:** Centralized seed distribution

Their findings indicated the combined heuristics of H1, H2, and H3 notably outperformed others, with H1 individually most effective. Despite valuable insights, their study lacked heuristic weighting and did not explore machine learning methods. Addressing these limitations, our research not only combines heuristic evaluations with Alpha-Beta pruning for enhanced performance but also explores advanced reinforcement learning methods such as Double DQN to further improve AI effectiveness.

[1] conducted experiments with various heuristics compared to standard and modified Minimax algorithms employing Alpha-Beta pruning. This study consisted of tournaments between human players and various algorithmic strategies, and algorithm-only round-robin matches. His results demonstrated the critical impact of initial move order on winning probability. Although heuristic methods struggled against human opponents, Minimax variants achieved significantly better outcomes, especially with increased search depth. The study further highlighted computational trade-offs, with heuristic methods requiring substantially less computation time compared to Minimax search algorithms. However, the approach [1] involved limited heuristic integration with Minimax, leaving significant scope for further improvements. Our study aims to advance this research by combining multiple heuristics with Alpha-Beta pruning to create a computationally efficient hybrid approach, alongside exploring sophisticated machine learning strategies for greater effectiveness.

B. Reinforcement Learning (RL)

Recent advancements in reinforcement learning (RL) have demonstrated significant potential in traditional board games, particularly in environments characterized by deterministic rules, turn-based dynamics, and sparse rewards. Although Mancala has received comparatively limited attention in the reinforcement learning literature, the game shares structural similarities with more extensively studied domains such as Go, Backgammon, and Othello. These games have served as testbeds for evaluating deep RL algorithms due to their strategic depth and the need for long-term planning.

Building on this foundation, the present study introduces a Double Deep Q-Network (Double DQN) agent for the Kalah variant of Mancala, enhanced with Prioritized Experience Replay (PER) and a three-stage curriculum learning framework. This design is intended to promote efficient policy convergence, strategic diversity, and long-term planning. By combining stable value approximation techniques with structured reward shaping, the agent learns to optimize actions across varying board configurations and game phases. In doing so, this work contributes to bridging the gap between deep reinforcement learning research and its application to lesser-studied strategic games like Mancala.

C. Research Objective

The primary objective of this research is to evaluate and compare the effectiveness of traditional search-based and modern learning-based AI strategies in the game of Mancala, specifically within the Kalah variant. Prior studies have emphasized classical approaches such as Minimax and basic heuristics. Meanwhile, this study aims to evaluate and compare the performance of two advanced AI strategies for the game of Mancala (Kalah variant): an **Advanced Heuristics agent** and a **Double Deep Q-Network (Double DQN)** agent enhanced with **Prioritized Experience Replay (PER)** and **curriculum learning**.

Specifically, the research seeks to:

1) Examine the impact of domain-informed heuristics on Alpha-Beta search efficiency and decision quality.

2) Assess the learning performance and generalization of a Double DQN agent trained with structured reward signals.

3) Benchmark both agents against baseline strategies including Random, Simple Minimax, and standard Alpha-Beta.

4) Investigate how reward shaping and curriculum design influence policy convergence in deterministic, turn-based environments.

Through systematic simulation and analysis, the study contributes a dual-framework approach that integrates classical search techniques with modern deep reinforcement learning.

III. METHODOLOGY

This section presents the design and implementation of various Mancala-playing agents, followed by the simulation framework used to evaluate their relative performance. The agents span from baseline strategies to advanced search-based and deep reinforcement learning techniques. Each strategy is described in detail, followed by an explanation of the evaluation protocol.

A. Random Strategy

The Random strategy serves as a control baseline. It selects moves uniformly at random from the set of legal actions available at each turn. While this approach lacks strategic foresight, it provides a useful reference point for evaluating the effectiveness of more sophisticated methods. The random agent is simple to implement and computationally efficient, but performs poorly due to its lack of situational awareness [3].

B. Simple Minimax

The Simple Minimax agent employs the classic adversarial search strategy by recursively evaluating game states to a fixed depth, assuming both players act optimally [3]. Each game state is assessed using a linear evaluation function that computes the pit score difference from the perspective of the maximizing player.

To accommodate Mancala-specific dynamics, the implementation handles extra turns by preserving the current player in the recursion when a move lands in their store. This introduces non-uniform tree depth, as branches may extend when the player retains control. The algorithm also supports flexible depth control by decrementing only when the turn changes.

The recursive function evaluates all valid moves, switching between maximizing and minimizing logic based on the current player. Importantly, the evaluation function is always invoked from the maximizing player's perspective to maintain consistency.

A simplified logic for depth-based recursion is:

```
if current_player == maximizing_for:
   # Maximizing the branch
   for move in valid_moves:
      new_board, extra_turn = make_move(...)
      next_player = current_player if extra_turn else other_player
      eval_score, _ = simple_minimax(new_board, depth - 1,
next_player, maximizing_for)
      ...
else:
   # Minimizing branch (# Invert score because we are in a
minimizing branch)
   ...
   eval_score = -eval_score
```

While the agent captures basic strategic planning, it lacks pruning mechanisms. As a result, its depth is limited by the exponential branching factor, leading to increased runtime and reduced responsiveness in real-time gameplay scenarios.

C. Minimax with Alpha-Beta Pruning

To address the inefficiency of exhaustive minimax search, alpha-beta pruning was incorporated. This enhancement introduces two dynamic thresholds: ***alpha*** (the best guaranteed score for the maximizer) and ***beta*** (the best

guaranteed score for the minimizer), to eliminate unpromising branches [17].

When the algorithm determines that a branch cannot influence the outcome (for example, when $\alpha \geq \beta$), it prunes the subtree, thereby reducing the number of evaluated nodes and enabling deeper searches within the same time budget.

This optimization is particularly impactful in Mancala, where extra turns can lead to extended decision chains. The pruning logic is carefully integrated to preserve the semantics of the game: when a player earns an extra turn, the recursion continues without switching roles, and depth remains unchanged for that call.

The structure mirrors Simple Minimax but includes the pruning condition:

```
if alpha >= beta:
    break  # prune remaining branches
```

D. Advanced Heuristics

The Advanced Heuristics agent integrates domain-specific knowledge into a depth-limited, alpha-beta pruned search [18]. Unlike traditional search strategies that rely solely on pit scores, this agent uses a handcrafted evaluation function that captures nuanced tactical and strategic patterns observed in expert Mancala gameplay.

At the heart of the agent lies the *heuristic_move_value*() function, which scores legal moves across **ten strategic dimensions**. Each feature reflects core aspects of optimal play, inspired by heuristic design principles in classical two-player games [3], [4]:

1. **Immediate Rewards**: Prioritizes moves that grant extra turns or enable captures—high-value plays in Mancala [5].
2. **Positional Advantage**: Emphasizes control of central pits, improving mid-game distribution and flexibility [6].
3. **Game Phase Awareness**: Adjusts priorities dynamically between early-game pit control and endgame score maximization [7].
4. **Defensive Play**: Penalizes moves that expose pits to capture and rewards those that block opponent tactics [8].
5. **Future Mobility**: Favors moves that preserve multiple future options, preventing early stagnation [9].
6. **Opponent Denial**: Discourages moves that enable opponent extra turns or easy captures [10].
7. **Seed Conservation**: Avoids depleting high-potential pits unless scoring benefits are significant [3].
8. **Tempo Control**: Rewards disruptive plays that create distribution complexity, making it harder for the opponent to plan ahead [6].
9. **Capture Vulnerability Mitigation**: Reduces the value of moves that place seeds into easily capturable locations [5].
10. **Balanced Distribution**: Encourages evenly spread seeds to maintain offensive and defensive versatility [7].

These scores are not only used to evaluate terminal states but also guide **move ordering** within alpha-beta pruning. Moves are sorted in descending heuristic value before expansion:

```
sorted_moves = sorted(valid_moves,
                      key=lambda m:
heuristic_move_value(board, m, maximizing_for), reverse=True)
```

This ordering improves pruning efficiency by evaluating promising moves first, increasing the likelihood of early cutoff [8], [11].

The implementation maintains **turn-based depth control**: when a move results in an extra turn, the depth is not reduced, allowing deeper lookahead on high-impact plays. Evaluations are consistently computed from the maximizing player's perspective, ensuring uniformity across branches [10].

By combining expert-informed heuristics with classical pruning techniques, the Advanced Heuristics agent effectively balances **strategic depth** and **computational feasibility**. It consistently outperforms traditional Minimax and Alpha-Beta agents in simulation, particularly in mid- and late-game phases where tactical precision is most critical.

E. Deep Q-Network (DQN)

The Deep Q-Network (DQN) agent implemented for Mancala leverages a model-free reinforcement learning paradigm to approximate optimal Q-values, generalizing decision-making across diverse game states. The agent's architecture and training procedures were significantly enhanced by integrating prioritized experience replay (PER), refined reward shaping, and curriculum learning. Training alternates between Player 1 and Player 2 positions to mitigate

first-move bias and encourage symmetric policy development [12].

1) Network Architecture and Optimization

The implemented neural network features three fully-connected layers with units configured as (64, 64, 32), interspersed with ReLU activations and a dropout rate of 10% to mitigate overfitting [12]. The network outputs six action-specific Q-values corresponding to the available pit selections. Training utilizes Mean Squared Error (MSE) loss with gradient clipping (norm capped at 1.0), optimized using the Adam optimizer with a learning rate of 0.0001. A target network, updated using Polyak averaging with a soft update coefficient $\tau = 0.015$, provides training stability by reducing oscillations in Q-value estimation [13].

2) Prioritized Experience Replay

The agent employs a Prioritized Experience Replay buffer, assigning sampling probabilities proportional to transition priorities based on Temporal Difference (TD) errors raised to an exponent $\alpha=0.6$ [14]. Importance sampling weights, which adjust for the sampling bias, anneal from $\beta=0.4$ to 1.0 over 100,000 training steps [14]. This mechanism emphasizes learning from impactful experiences, accelerating policy convergence [14].

3) State Representation

Each Mancala state is encoded as a 29-dimensional normalized feature vector comprising:

- Player and opponent pit and store values
- Relative differences between pits and stores
- Turn indicators and game phase indicators
- Strategic indicators such as seed distribution, pit vulnerability, and potential tactical moves

This comprehensive feature set allows nuanced and informed decision-making, adaptable to both player roles [15].

4) Reward Shaping and Curriculum Learning

The reward function underwent extensive reshaping to encourage strategically diverse gameplay. The best-performing model used the following reward shaping scheme:

- Store gain this turn
- Store differential (weighted 0.8)
- Terminal win/loss reward awarded at the end of the game
- Extra turn (bonus 1.4)
- Capture actions (bonus 0.7)
- Potential captures and extra turns (bonus 0.2 each)
- Defensive moves preventing opponent advantage (bonus 0.15)
- Tempo control and seed conservation heuristics

The curriculum learning framework is structured into three progressive phases, each designed to introduce greater strategic complexity incrementally [8], [16]. In **Phase 1**, the agent is trained using a foundational reward function emphasizing fundamental gameplay mechanics—namely store gain, terminal win/loss conditions, extra turns, and successful capture events. This stage aims to build baseline competencies in scoring and turn retention.

Phase 2 introduces additional tactical dimensions by incorporating heuristic-based rewards for potential captures, anticipated extra turns, and the prevention of opponent advantages through defensive positioning. This enables the agent to begin developing anticipatory strategies and situational awareness.

In **Phase 3**, the reward function is further refined to include penalties for depleting key resource pits (seed conservation), bonuses for disruptive seed distributions (tempo control), and negative incentives for exposing pits to potential captures (capture vulnerability mitigation). These enhancements promote long-term planning and positional optimization.

Transitions between phases are governed by a fixed episode schedule, with shifts occurring after 10,000 and 20,000 training episodes, respectively. This phased approach facilitates a structured learning trajectory, enabling the agent to first internalize core gameplay rules before advancing toward more sophisticated, high-level decision-making strategies. This phased reward design supports stable convergence and progressive acquisition of strategic depth, in line with curriculum learning principles established by [16].

5) Empirical Improvements

Empirical results (**Appendix E**) indicate significant performance improvements attributed to reward reshaping and curriculum learning. Training trajectories demonstrated steady improvements in win rates through 24,000 episodes, achieving an approximately 70%-win rate against baseline opponents after integrating these methodologies. The baseline opponent is randomly chosen every episode between a Random agent and a Minimax agent. This iterative training approach, demonstrated in the performance charts, shows marked enhancements in both stability and strategic diversity compared to initial baseline models. These

improvements align with findings in reinforcement learning literature that highlight the benefits of combining PER, strategic reward shaping, and staged learning curricula [8], [14], [16].

Further insights into the effectiveness of **reward shaping** and **curriculum learning** are illustrated in **Appendix E**, which tracks win rate trajectories across different training configurations. The DQN model using both the full proposed reward function and the three-phase curriculum (red line) achieved superior and more stable convergence, culminating in the observed 80.8% win rate (result shown in the Section IV) against the Random agent. In comparison, the version trained using only the Phase 1 reward function (yellow line) and the version using the complete reward function without curriculum learning (blue line) underperformed both in convergence speed and final win rate. These results highlight the critical role of progressive training structure and strategically engineered rewards in enabling deep reinforcement learning agents to learn nuanced, high-level strategies in deterministic environments like Mancala.

The combination of advanced reward engineering and incremental difficulty scaling via curriculum learning has thus proven essential in achieving robust and generalizable AI performance in Mancala.

IV. Results and Analysis

This section presents the results of large-scale simulations conducted to evaluate the performance of six Mancala-playing agents: Random, Simple Minimax, Minimax with Alpha-Beta Pruning, Advanced Heuristics, and a Deep Q-Network (DQN) agent. Each agent was evaluated against all others in both Player 1 and Player 2 positions, and win rates were aggregated to assess strategic effectiveness. Overall, the data confirms that strategic depth, search efficiency, and domain-informed heuristics are key determinants of success in the Mancala environment.

In total, 12,000 games were simulated to evaluate the performance of five Mancala-playing AI strategies: Advanced Heuristic, Deep Q-Network (DQN), Minimax with Alpha-Beta pruning, Simple Minimax, and Random. The simulations included 24 unique strategy pairings, with each strategy evaluated comprehensively both as the first player (Player 1) and as the second player (Player 2). The performance metrics, including win rates and computational efficiency, were analyzed to determine the effectiveness of each strategy. Detailed win rate heatmaps of each strategy (Simple Minimax, Alpha-Beta Pruning, Advanced Heuristics, Deep Q-Network) are provided in **Appendix A, B,** and move distribution is provided in **Appendix Section C**.

A. Win Rate Analysis

The performance in terms of win rates was visualized through heat maps presented in Appendices A and B. When acting as Player 1, who has the inherent advantage of making the initial move, the Advanced Heuristics strategy dominated clearly, securing near-perfect victory rates (100%) against almost all other strategies (Appendix A). The Minimax Alpha-Beta and Simple Minimax strategies similarly exhibited strong performances against weaker strategies, especially Random and Simple Minimax, but struggled significantly against the Advanced Heuristics strategy.

The DQN strategy showcased notable competitive strength, achieving high win rates against Random (80.8%), Simple Minimax (42%), and Minimax Alpha-Beta (77.4%), but performing moderately against the Advanced Heuristics (46.8%). Conversely, the Random strategy generally underperformed, as expected due to its lack of strategic foresight, infrequently surpassing a 50%-win rate.

As **Player 2** (**Appendix B**), **win rates** declined noticeably across all strategies due to the inherent disadvantage of moving second. Both the Simple Minimax and Minimax Alpha-Beta agents experienced substantial performance drops, often nearing zero-win rates when matched against more advanced opponents. In contrast, the Advanced Heuristics agent maintained strong performance, achieving over 90% win rates against all other agents, except the DQN, against which it registered a more balanced outcome of 50%. Notably, the DQN agent demonstrated considerable robustness, maintaining moderate success even from less favorable positions, underscoring its adaptability and capacity to generalize across different gameplay conditions.

B. Move Distribution

The **move distributions** (**Appendix C**) varied significantly across strategies, reflecting differing strategic complexities. The Random strategy exhibited the highest median number of moves, indicative of its inefficiency and lack of targeted strategy. Simple Minimax and Minimax Alpha-Beta displayed lower median moves, indicative of more strategic precision and effective decision-making,

leading to faster game conclusions. However, the Simple Minimax agent occasionally incurred **substantial computational overhead**. This is due to its exhaustive search nature, which incurs a time complexity of $O(b^d)$ and space complexity of $O(bd)$.

Advanced Heuristics and DQN strategies demonstrated intermediate move distributions, reflecting a balance between strategic complexity and execution efficiency. Particularly, the DQN strategy displayed a narrower range of moves, suggesting consistent and stable decision-making across various gameplay scenarios.

C. Computational Efficiency

The computational efficiency of each strategy was evaluated based on total execution time per game, combining results from both Player 1 and Player 2 perspectives (Appendix D). As shown in the boxplot, the DQN agent was the most efficient, consistently producing sub-second decision times with minimal variance. The Random strategy also exhibited low execution times, although with slightly higher variability due to stochastic move selection logic.

In contrast, the Simple Minimax agent showed noticeably higher and more variable execution times. This is attributable to its brute-force traversal of the game tree. With a time complexity of $O(b^d)$, where b is the branching factor and d is the search depth, Simple Minimax can become computationally expensive, especially in Mancala, where extra turns introduce non-uniform recursion depth.

The Minimax Alpha-Beta strategy demonstrated moderate execution times, benefiting from pruning unpromising branches, which effectively reduces the average-case complexity to approximately $O(\frac{bd}{2})$. Meanwhile, the Advanced Heuristics agent had the highest execution time in several instances, with outliers exceeding 10 seconds. This overhead is likely due to the computational cost of evaluating and ranking multiple strategic features for every valid move before expansion. These operations, though beneficial for strategic depth, can be expensive, especially when additional lookaheads or simulations are embedded within the heuristic function.

In addition to simulation-based evaluation, a fully functional **web application (Appendix F)** was developed to visualize and interact with the Mancala game and AI agents. The interface supports human-versus-AI gameplay and agent-versus-agent demonstrations, enabling real-time inspection of decision-making processes and gameplay dynamics. This tool not only facilitated debugging and demonstration during development but also lays the groundwork for future human-AI evaluation studies and educational use.

Overall, while the Advanced Heuristics agent offers strong performance, its execution time may pose challenges in real-time applications. The DQN agent, by contrast, provides an excellent balance between speed, consistency, and adaptability, making it particularly well-suited for scalable and generalizable gameplay scenarios.

V. CONCLUSION

Detailed quantitative results and further breakdown of data visualizations are available in Appendices A through D.

This study presented a comprehensive evaluation of AI strategies for the game of Mancala, focusing on the Kalah variant. Two advanced agents were developed and compared: a heuristic-driven Alpha-Beta pruning agent and a Double Deep Q-Network (Double DQN) agent enhanced with Prioritized Experience Replay (PER) and curriculum learning. Through large-scale simulations against a diverse set of baseline agents, the Advanced Heuristics agent demonstrated superior tactical precision, particularly when acting first, while the DQN agent exhibited strong generalization, consistent performance, and computational efficiency across varied gameplay scenarios.

The findings highlight the complementary strengths of search-based and learning-based approaches: heuristic search offers high-performance play through expert-crafted evaluation functions, whereas reinforcement learning provides adaptability and scalability through autonomous policy development. Notably, the curriculum-guided DQN agent achieved a competitive win rate of approximately 70% within just 24,000 training episodes.

Limitations and Future Work:

While the findings are promising, several **limitations** should be acknowledged. **First**, the DQN agent was trained for a limited duration (24,000 episodes) due to time constraints. Extended training could improve its strategic complexity, particularly in endgame scenarios and infrequent board states. **Second**, computational resource limitations constrained both training throughput and search depth for Alpha-Beta agents. More powerful hardware or optimized implementations could enable deeper simulations and larger

batch updates, enhancing both agent types. **Third**, this study was restricted to a single rule variant (Kalah). Mancala encompasses a diverse family of games, and the agents' ability to generalize across variants such as Oware, Bao, or Sungka remains unexplored.

Several future directions remain for extending and enhancing the work presented in this study. First, the performance of the Deep Q-Network (DQN) agent could be further improved through extended training durations, the use of more powerful GPUs to accelerate training cycles, and expansion of the replay buffer size. These improvements would allow the agent to experience a broader and more diverse range of game states, facilitating deeper strategic learning, particularly in complex or rare late-game scenarios.

Second, future research may explore the use of **transfer learning** or **multi-task reinforcement learning** to enable generalization across different Mancala variants. Such approaches could help agents adapt to new rulesets or board configurations with limited retraining, increasing their flexibility and practical applicability in broader game settings such as Oware, Sungka, or Bao.

Third, there is an opportunity to incorporate **neural-symbolic reasoning** or **explainable AI (XAI)** techniques to improve the interpretability of agent decisions. By making agent policies more transparent, researchers and users alike could gain deeper insights into the rationale behind critical moves, which is particularly valuable for educational tools, debugging, and ethical AI applications.

Additionally, a **web-based Mancala application** was developed as part of this project to support interactive testing and real-time visualization of agent behavior. This application can be extended in future work to support online reinforcement learning, human-in-the-loop training, or adaptive difficulty adjustment. Such enhancements would enable dynamic, human-centered evaluation and promote real-world usability in educational or recreational contexts.

Together, these directions aim to expand the scalability, generalizability, and transparency of Mancala-playing agents while bridging the gap between simulation-based research and interactive, human-facing applications.

In summary, this research contributes a dual-framework methodology, combining classical heuristic search and modern deep reinforcement learning, for building high-performing, interpretable, and adaptive AI agents in deterministic, turn-based games.

VI. REFERENCES

[1] L. Pekař, R. Matušů, J. Andrla, and M. Litschmannová, "Review of Kalah Game Research and the Proposition of a Novel Heuristic–Deterministic Algorithm Compared to Tree-Search Solutions and Human Decision-Making," *Informatics*, vol. 7, no. 3, p. 34, Sep. 2020, doi: 10.3390/informatics7030034.

[2] C. Gifford, J. Bley, D. Ajayi, and Z. Thompson, "Searching and Game Playing: An Artificial Intelligence Approach to Mancala," ITTC-FY2009-TR-03050-03, Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, Jul. 2008.

[3] Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[4] Pearl, J. (1984). Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley.

[5] Campbell, M., Hoane, A. J., & Hsu, F. H. (2002). Deep Blue. Artificial Intelligence, 134(1-2), 57–83.

[6] Browne, C., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., ... & Colton, S. (2012). A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 1–43.

[7] Müller, M. (2002). Computer Go. Artificial Intelligence, 134(1-2), 145–179.

[8] D. E. Knuth and R. W. Moore, "An Analysis of Alpha-Beta Pruning," Artificial Intelligence, vol. 6, no. 4, pp. 293–326, 1975.

[9] Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in UCT. In Proceedings of the 24th International Conference on Machine Learning (pp. 273–280).

[10] Von Neumann, J., & Morgenstern, O. (1944). Theory of Games and Economic Behavior. Princeton University Press.

[11] Schaeffer, J. (1989). The history heuristic and alpha-beta search enhancements in practice. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(11), 1203–1212.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.

[13] T. P. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized Experience Replay. International Conference on Learning Representations (ICLR).

[15] MarcGBellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In International Conference on Machine Learning, pages 449–458. PMLR, 2017

[16] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. Proceedings of the 26th Annual International Conference on Machine Learning (ICML), 41–48.

[17] S. Narvekar, J. Peng, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," Journal of Machine Learning Research, vol. 21, no. 181, pp. 1–50, 2020.

[18] J. Schaeffer, "The History Heuristic and Alpha-Beta Search Enhancements in Practice," IEEE Trans. Pattern Anal. Mach. Intell., vol. 11, no. 11, pp. 1203–1212, 1989.

[19] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," nature, vol. 529, Art. no. 7587, 2016.

VII. Appendix

A. *Win Rate Heatmap – Player 1*

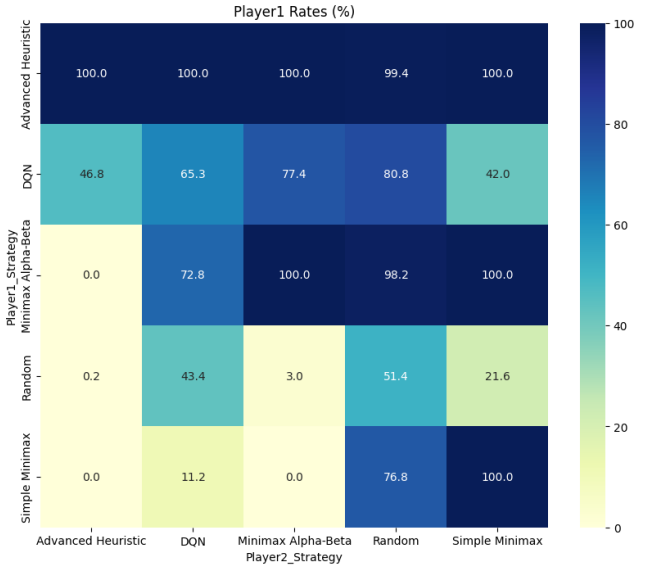


Fig. 4. Win rate heatmap for Player 1 positions across all agent matchups.

B. *Win Rate Heatmap – Player 2*

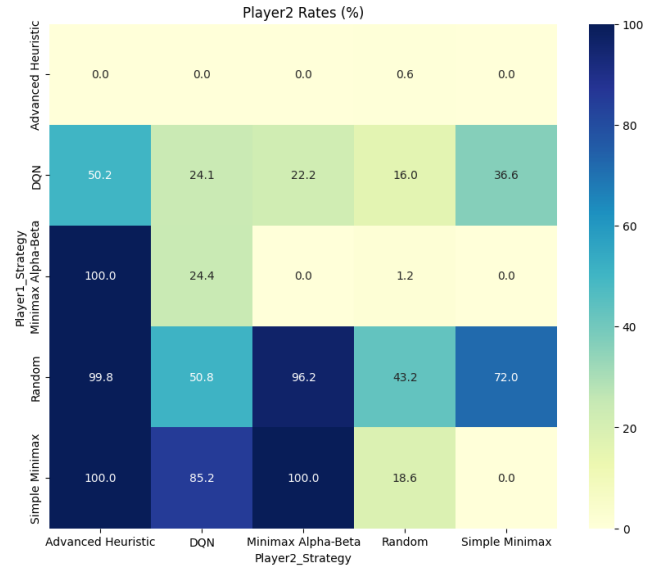


Fig. 5. Win rate heatmap for Player 2 positions across all agent matchups.

C. *Move Distribution Analysis*

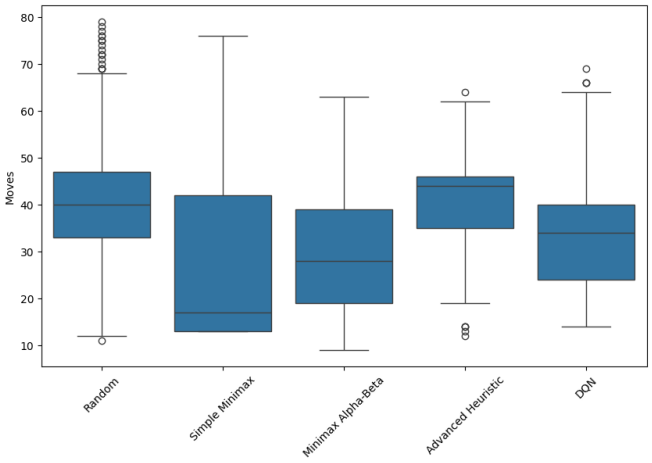


Fig. 6. Move Distribution by Strategy

D. *Execution Time Distribution*

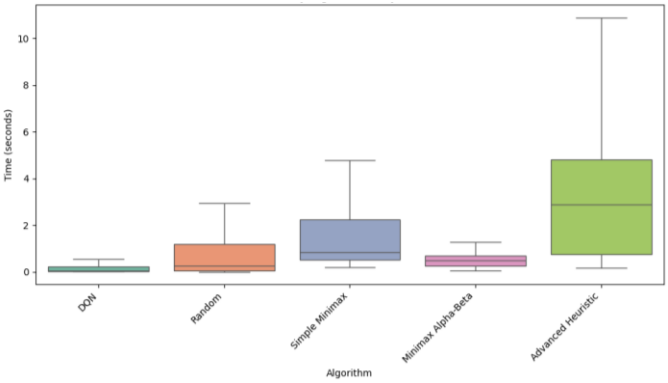


Fig. 7. Time Distribution by Strategy

E. DQN Reward Function Comparison

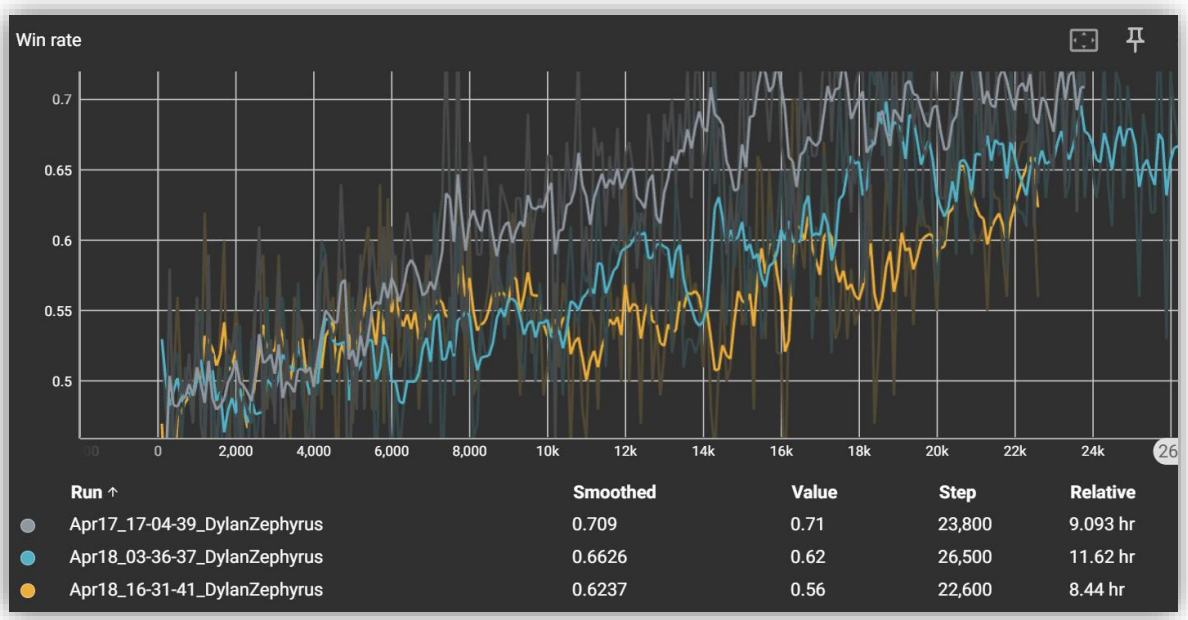


Fig. 8. Win Rate Progress During Training of 3 Deep Q-Network (Different Reward Functions)

F. Mancala Web Application UI

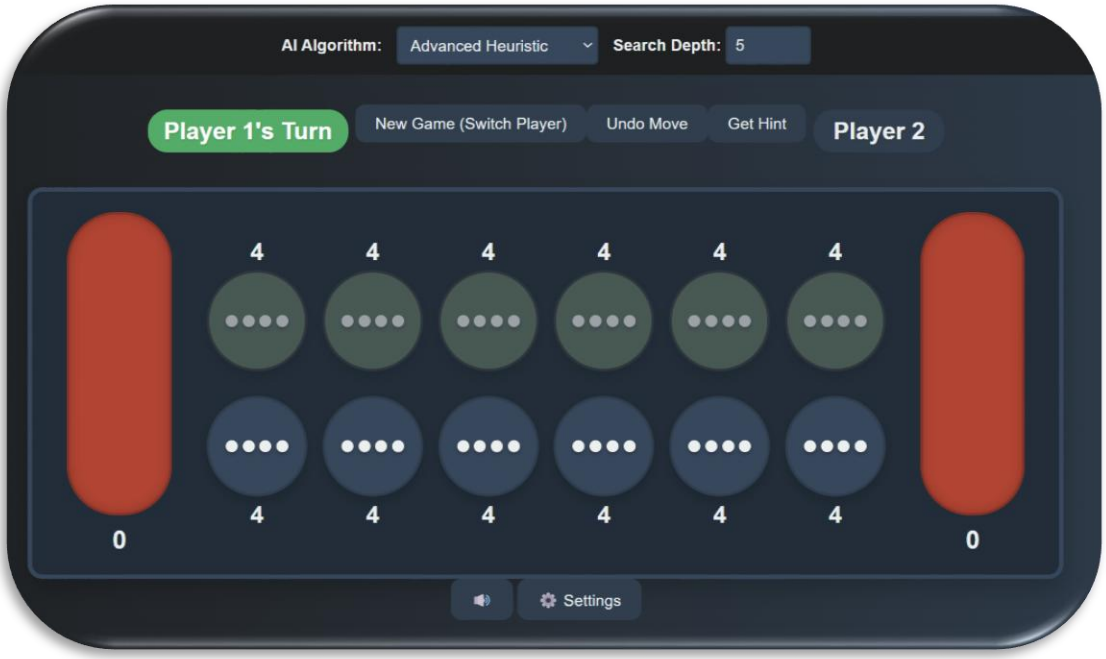


Fig. 9. Mancala Web Application UI