Reinforcement Learning for Quoridor

Anonymous Authors

***Abstract*—This report explores different reinforcement learning approaches applied to the Quoridor board game. We analyze classical methods such as Minimax and Monte Carlo Tree Search (MCTS) and evaluate their effectiveness on different board sizes and game configurations. We also propose modifications to heuristics, particularly a path obstruction heuristic for MCTS, and learning strategies to enhance agent performance. Our implementation and evaluation of the path obstruction heuristic demonstrate significant improvements in the AI's strategic behavior and decision-making, better aligning the agent's play with the fundamental principles of Quoridor strategy.**

I. Introduction

Quoridor is a turn-based strategy game for 2 to 4 players in which players must reach the opposite side of the board of an odd number of tiles while placing barriers to slow down their opponents. The game presents a challenging decision-making problem that can be tackled using reinforcement learning techniques. Previous works have attempted to solve Quoridor using Minimax, Monte Carlo Tree Search (MCTS), and Deep Q-Networks (DQN). This project is particularly suitable for reinforcement learning because it has simple mechanics (moving or placing a wall), but the complexity of the state space grows exponentially, and with no consensual heuristic, it leaves plenty of space to explore. The complexity primarily lies in the placement of walls, which makes calculating every possibility tedious and often computationally unfeasible.

We took inspiration from previous research on Quoridor AI [1], where the results were somewhat inconclusive due to time and resource constraints. Our goal is to continue this work and develop an agent that reliably outperforms simple strategies. This project aims to extend previous work by:

1) Testing various algorithms on different board sizes and with multiple players
2) Modifying heuristics to better capture the strategic essence of Quoridor
3) Implementing intermediate rewards to guide learning

As part of this project, we have implemented a significant enhancement to the MCTS algorithm: a path obstruction heuristic that evaluates moves based on how effectively they increase the opponent's path length to the goal. This heuristic aligns with the core strategic principle of Quoridor—forcing opponents to take longer paths while optimizing one's own path.

Our implementation demonstrates that incorporating domain-specific knowledge into general-purpose algorithms significantly improves AI performance in terms of both strategic play and alignment with human-like decision making in Quoridor.

II. Background

Previous research on Quoridor AI has largely focused on two-player games on small grids (e.g., 3×3 or 5×5). The most promising method, MCTS, balances exploration and exploitation to determine optimal moves. Some studies have also combined MCTS with heuristic functions to further refine decision making. This method seems promising mainly because it was used to solve the game of Go with great success, and given that it is a similar problem (perfect information, turn-by-turn board game with a limited set of moves), it was deemed the appropriate approach [1].

For a square board game of size n, we have:

- Player positions $p_1, p_2 \in \{1, ..., n\}^2$
- Wall positions $W \subset \{(x, y, o) | x, y \in \{1, ..., n-1\}, o \in \{h, v\}\}$ where h and v represent horizontal or vertical wall orientation
- Remaining walls for each player $w_1, w_2 \in \mathbb{N}$ depending on the number of walls allocated to each player

The action space is:

- Movement actions $A_m = \{N, S, E, W, NE, NW, SE, SW\}$
- Wall placement actions $A_w = \{(x, y, o) | (x, y) \in \{1, ..., n-1\}, o \in \{h, v\}\}$
- The space of legal moves $A_{legal} \subset A_m \cup A_w$

Quoridor is deterministic, so there are no probabilities of transition after taking an action a. We denote $R(s, a, s')$ as the reward function when an action a makes the agent move from state s to state s'. We also denote $V^\pi(s)$ as the state-value function and $Q^\pi(s, a)$ as the action-value function. Finally, we have $\tau = (s_0, a_0, r_1, s_1, a_1, ..., s_k, a_k, r_{k+1})$ representing the trajectory.

A key concept in Quoridor strategy is the calculation of shortest paths. We define $SP(p, W)$ as the function that calculates the shortest path length for a pawn at position p to reach its goal, given the wall configuration W. This calculation typically employs breadth-first search or Dijkstra's algorithm.

III. Methodology

A. Agent and Learning Strategies

We use the same base environment to evaluate each method, creating agents that apply different algorithms for comparison. When creating our environment, we specify the board size, the number of walls allocated to each player (the same for all players), and the opponent type.

Our approach involves implementing and comparing the following methods:

- **Minimax Algorithm**: A classical game-theoretic approach that maximizes the agent's gains while minimizing

the opponent's advantage by evaluating future possible game states.
- **Deep Q-Networks (DQN)**: A reinforcement learning approach that uses a neural network to approximate Q-values.
- **Monte Carlo Tree Search (MCTS)**: The most promising method, leveraging simulation-based exploration to find optimal moves.

We evaluate these methods on different board sizes (5×5, 9×9) and explore enhancements such as improved heuristics and intermediate rewards.

B. Path Obstruction Heuristic for MCTS

The core innovation of our project is the implementation of a path obstruction heuristic that enhances the MCTS algorithm for Quoridor. This heuristic specifically targets the move selection process by incorporating domain knowledge about the strategic importance of increasing the opponent's path length.

The standard MCTS algorithm selects moves based primarily on the number of simulations and win rates. Our modified algorithm enhances this selection process by:

1) Simulating each potential move to evaluate its impact on the opponent's path
2) Calculating the increase in the opponent's path length resulting from each move
3) Assigning additional value to moves that maximize this increase
4) Selecting moves based on this adjusted evaluation rather than raw simulation counts

The key modification is in the 'selectBestMove()' function of the MCTS implementation:

Algorithm 1 Path Obstruction Heuristic for MCTS

selectBestMove**for** each child c in root.children **do**
simulationGame ← getSimulationGameAtNode(c)
opponentPathBefore ← getShortestDistanceToGoalFor(game.pawnOfNotTurn, game)
opponentPathAfter ← getShortestDistanceToGoalFor(simulationGame.pawnOfTurn, simulationGame)
pathIncrease ← opponentPathAfter - opponentPathBefore
c.adjustedNumSims ← c.numSims + max(0, pathIncrease) × 5
end for
maxAdjustedSimsIndex ← argmax(root.children[i].adjustedNumSims)
best ← root.children[maxAdjustedSimsIndex]
return move: best.move, winRate: best.winRate

The coefficient (5) was determined through empirical testing and represents the weight given to path obstruction relative to the statistical confidence provided by simulation counts.

C. Implementation Details

We implemented the Python version of Quoridor with Pygame for our main experiments with various reinforcement learning approaches. For the path obstruction heuristic, we built upon an existing JavaScript implementation of Quoridor, modifying its MCTS algorithm to incorporate our heuristic.

The complete implementation includes:

- Breadth-first search for calculating shortest paths
- Standard MCTS implementation with UCT selection
- Our path obstruction heuristic integrated into the move selection process
- Random playouts for simulation phase

Our implementation is accessible at [2].

IV. Experiments and Results

A. Planned Experiments

Our experimental approach includes:

- Implementing baseline agents (random, shortest-path greedy) for benchmarking
- Introducing variations such as random barrier placements and multiple players
- Comparing the Minimax, DQN, and MCTS algorithms on standard settings
- Analyzing learning efficiency and convergence rates

B. Path Obstruction Heuristic Evaluation

We have completed the implementation and evaluation of our path obstruction heuristic for the MCTS algorithm. The results demonstrate significant improvements over the standard MCTS implementation.

1) Strategic Behavior: The enhanced AI demonstrates markedly improved strategic wall placement. While the standard MCTS would often place walls reactively or prioritize pawn movement, our modified algorithm consistently places walls in positions that maximize the opponent's detour.

Qualitative analysis of gameplay shows that the enhanced AI:

- Places walls to create longer detours rather than merely blocking immediate paths
- Demonstrates more human-like strategic thinking
- Better balances progression and obstruction
- Creates more complex and challenging gameplay

2) Decision-Making Alignment: Our heuristic successfully aligns the AI's decision-making with Quoridor's core mechanics. The algorithm now prioritizes moves that force the opponent to take longer paths while maintaining its own progress toward the goal.

This improved alignment manifests in several ways:

- More frequent wall placements in the middle game
- Strategic wall configurations that create compound detours
- Improved adaptability to the opponent's strategy
- Better utilization of limited wall resources

3) Equilibrium Between Exploration and Exploitation: The path obstruction heuristic maintains the exploration capabilities of MCTS while enhancing exploitation of domain knowledge. This balance is crucial for achieving strong play without sacrificing the algorithm's ability to discover novel strategies.

V. Conclusion and Future Work

Our project demonstrates the effectiveness of enhancing general reinforcement learning algorithms with domain-specific knowledge for the game of Quoridor. In particular, our path obstruction heuristic significantly improves the MCTS algorithm's strategic behavior, creating an AI that better captures the essence of skilled Quoridor play.

Current limitations of our implementation include:

- The static weighting coefficient (5) does not adapt to different game phases
- The algorithm only considers immediate path increases, not long-term strategic positions
- No consideration of the "time" factor or remaining moves
- Limited by the computational constraints of the MCTS framework

For future work, we recommend several promising directions:

- **Dynamic parameter adjustment**: Automatically adjust the coefficient based on game phase (opening, middle, endgame)
- **Multi-move analysis**: Extend the evaluation to consider the impact of moves several steps ahead
- **Reinforcement learning integration**: Combine or replace MCTS with neural networks trained through reinforcement learning (DQN or AlphaZero-style approaches)
- **Opponent modeling**: Incorporate mechanisms to identify and exploit patterns in opponent play
- **Simulation optimization**: Reduce calculation time per simulation to allow more iterations within the same time constraints
- **Complex position evaluation**: Incorporate additional evaluation factors such as center control, option flexibility, and the ratio of remaining walls

In conclusion, our project contributes to the field of AI for Quoridor by:

1) Demonstrating the importance of domain-specific heuristics in enhancing general-purpose algorithms
2) Providing a concrete implementation of a path obstruction heuristic that significantly improves MCTS performance
3) Outlining a roadmap for future research in this domain

Our work shows that even modest modifications to reinforcement learning algorithms, when informed by domain knowledge, can substantially improve AI performance in complex strategic games.

References

[1] C. Jose et al., "Quoridor AI Research Paper," 2023. Available: https://www.researchgate.net/publication/363456787_Quoridor-AI. [Accessed: 2025-03-04].

[2] ANONYMOUS, "Our source code," https://github.com/LaComete3/Quoridor_581, 2025.

[3] deepanshut041, "Reinforcement-Learning," https://github.com/deepanshut041/Reinforcement-Learning, 2020.

[4] samkovaly, "quoridor-game-AI," https://github.com/samkovaly/quoridor-game-AI, 2020.

[5] C. B. Browne et al., "A Survey of Monte Carlo Tree Search Methods," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1-43, March 2012.

[6] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," MIT Press, 2018.

[7] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science, vol. 362, no. 6419, pp. 1140-1144, 2018.