Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/73/20240390

CCC--Uno AI agent auxiliary system

Chi Zhang[1,3,*], Zhengxiao Chen[2,4]

[1]Delaware Institute of Data Science, South West University of Finance and Economics, Chengdu, 610072, China
[2]School of Computer and Information, Anhui Normal University, Anhui, 241000, China

[3]cz8140163@gmail.com
[4]ethanchenczx@foxmail.com
*corresponding author

Abstract. In this paper, we explore the design and implementation of a UNO game AI Agent based on the Mini-Max algorithm, and we named it "CardCraft Champ". UNO, as a popular card game, is filled with strategies and uncertainty, making it challenging to design an AI Agent that can efficiently handle different game scenarios. Our goal is to develop an AI system that can intelligently select cards, respond to opponents' strategies, and achieve victory in UNO games. The core idea of "CardCraft Champ" is to use the Mini-Max algorithm and Alpha-Beta pruning for decision-making. By conducting deep searches and evaluations on the game state, our AI Agent can predict various possible game progressions and make optimal decisions based on expected outcomes. The Alpha-Beta pruning technique can accelerate the search process even further, enabling our AI to find the optimal solution within a limited time and make wise decisions in the game. Our research not only focuses on the implementation of the algorithm but also on its application to actual UNO games. Through comparative experiments with other existing UNO game AIs, we demonstrate the superior performance of "CardCraft Champ" in different scenarios. The experimental results show that our AI Agent can minimize risks in the game while maintaining a high winning probability against various opponent strategies and game scenarios. Additionally, we consider the scalability and applicability of "CardCraft Champ". Our system performs well in battles of different difficulty levels, proving its adaptability and stability. We also explore possible directions for improvement, such as introducing deep learning techniques to enhance the AI's decision-making capabilities.

Keywords: UNO game, Mini-Max algorithm, Alpha-Beta pruning, evaluation function, decision tree.

1. Introduction

1.1. Background and challenge of the game UNO

As an ancient and widespread form of entertainment, card games have always been an important part of people's leisure activities. Among them, UNO game has become a popular choice for family gatherings and social occasions due to its simple and easy-to-learn rules, diverse gameplay, and intense competitiveness. UNO game was first introduced by Merle Robbins in the United States in 1971 and

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/73/20240390

quickly became a well-known classic card game worldwide. UNO game is popular for its unique rules and mechanics. Players match their cards to the cards in a discard pile, with the goal of getting rid of all their cards before other players. However, the game includes various special cards and reverse rules, which make it full of strategy and uncertainty. This combination of competition and strategy not only tests players' decision-making abilities but also their insight into opponents' intentions.

The UNO game also presents another challenge, which is the extremely complex decision space within the game [1]. In each round, players have the choice of which card to play, whether to use special cards or add one more card and skip the round, and these decisions will impact the progress and outcome of the game. Specifically, players need to dynamically adjust their decisions based on the current game state, their own hand, and the possible opponent strategies in order to maximize their advantage. This complex decision-making process requires AI agents to possess excellent decision-making abilities and game strategies.

In summary, UNO, as a competitive and strategic card game, not only brings joy to human players but also presents many challenges for artificial intelligence researchers. Designing an AI agent that can intelligently handle changing situations and complex game strategies in UNO is a thought-provoking problem. In this study, we will explore the design and implementation of a UNO game AI agent based on the Mini-Max algorithm and Alpha-Beta pruning, aiming to overcome these challenges and achieve excellent performance.

1.2. The Application of AI in Card Games

Since the birth of artificial intelligence technology, its application in various fields has gradually demonstrated tremendous potential. Among them, card games, as complex gaming environments, have attracted numerous researchers and developers to apply AI technology to achieve intelligent game agents. This trend has not only promoted the development of the card game industry but also provided practical scenarios for the improvement and innovation of artificial intelligence algorithms.

Prior to this, many AI agents have achieved impressive results in games. For example, AlphaGo, DouZero [2], and Suphx [3] have achieved unprecedented performance in the games of Go, Dou Dizhu and Mahjong. Recent research is moving towards more challenging games with incomplete information, where intelligent agents compete or cooperate with other agents in partially observable environments. Significant progress has been made, starting from two-player games, such as simple Leduc Hold'em and Texas Hold'em, and gradually expanding to multiplayer games such as multiplayer Texas Hold'em, StarCraft, DOTA, Hanabi, Mahjong, and Honor of Kings, achieving inspiring results

1.2.1. The Advantage of AI in Card Games. Card games, as one of the classic problems in game theory research, involve rich decision-making and tactics. The application of AI in card games mainly focuses on the following aspects:

1. **Game strategy optimization**: AI can learn the optimal strategies in card games to make the best decisions in different situations. For example, in poker games, AI agents can learn the best betting strategies for different hand combinations through deep learning methods, thereby enhancing their competitiveness in the game.
2. **Opponent modeling and** prediction: In card games, the behavior and strategies of opponents are crucial for decision-making. AI can analyze the opponents' historical actions and behavior patterns to build models for predicting their possible next moves, thus better responding to their strategies.
3. **Incomplete information games**: Card games often involve incomplete information, where players can only see part of the information. AI needs to utilize limited information to infer and predict other players' strategies and make reasonable decisions under incomplete information.
4. **Monte Carlo simulation**: Monte Carlo simulation [4] is a common method used to estimate optimal decisions in uncertain environments. AI can use Monte Carlo methods to simulate different possible scenarios in card games to find the best action strategies.

DOI: 10.54254/2755-2721/73/20240390

5. Intelligent game agents: AI is also applied in creating intelligent game agents in card games to provide interesting gameplay experiences with human players. These agents can adjust their strategies based on different difficulty levels, making the game more challenging and enjoyable.

1.3. The Objective And method of This Study

1.3.1. The Choice of the Algorithm in the AI. The Mini-Max algorithm is a common game tree search method widely used in games such as card games. This algorithm constructs a game tree by recursively simulating the decision-making process of the player and the opponent, and selects strategies to maximize our gains or minimize the opponent's gains at each level of the tree. However, the traditional Mini-Max algorithm faces an exponentially growing state space when the search depth is large, resulting in a huge computational burden and difficulty in finding the optimal solution within a practical time frame.

To address this issue, we introduce the Alpha-Beta pruning technique. Alpha-Beta pruning effectively reduces the search space by excluding some branches that do not need to be considered during the search of the game tree, thereby improving the search efficiency. This technique is based on a key idea: if we find that the value of a node exceeds the currently known best value range (Alpha and Beta), we can abandon the search for that node, thereby reducing unnecessary calculations. By combining Alpha-Beta pruning with the Mini-Max algorithm, we are able to find an approximate optimal playing strategy within a limited time frame, enabling the AI agent to make wise decisions in the UNO game.

1.3.2. Design of the Evaluation Function. In order to enable the AI agent to evaluate the merits and drawbacks of each possible card-playing strategy, we need to design a suitable evaluation function. The evaluation function plays a crucial role in the UNO game as it assigns a score to each card-playing strategy based on the current game state and the cards in the player's hand. This score reflects the potential benefits or risks of the strategy in the game progression.

Our evaluation function takes into account multiple factors, including the player's current hand combination, the effects of special cards, and the status of other players. For example, if a card-playing strategy allows the current player to quickly play all the cards in their hand and achieve victory, it will receive a higher score. On the other hand, if a card-playing strategy may trigger special card effects from opponents, forcing the current player to draw more cards, the score for that strategy will be relatively lower. By weighing these factors reasonably, our evaluation function can provide a reasonable reference score for each card-playing decision, thereby guiding the AI agent to make the best choice.

In summary, our research aims to develop an AI agent for the UNO game that is capable of intelligently playing cards and adapting to different game situations by applying the Mini-Max algorithm and Alpha-Beta pruning technique. By designing and implementing a suitable evaluation function, we enable the AI to consider multiple factors in the decision-making process and make optimal strategic choices

2. The Design and Implementation of "CardCraft Champ"

2.1. Game Rules and State Representations

In this section, we will provide a detailed introduction to the UNO game rules and state representation methods used in our research. To simplify the study, we have made certain reductions to the rules of the UNO game while retaining its core elements in order to facilitate the design and implementation of AI agents. ….

2.1.1. Simplified Description of the Game Rules. In our study, the rules of the UNO game were simplified to the following scenario: There are a total of 12 cards in the game, with four different colors - red, blue, yellow, and green. Each color has five different numbers - 0, 1, 2, 3, and 4. We have two

DOI: 10.54254/2755-2721/73/20240390

players, each starting with 3 cards. The goal of the game is to be the first player to get rid of all the cards in their hand to win.

In each player's turn, they can perform one of the following actions:

Action 1. Play a card with the same color as the previous card.

Action 2. Play a card with the same number as the previous card.

Action 3. If unable to perform actions 1 and 2, they need to draw a card from the deck.

The cards played will go into the discard pile, and the cards drawn will come from the deck. When there are no more cards in the deck, the discard pile will be shuffled and placed back at the bottom of the deck.

2.1.2. State Representation Method. In order for the AI agent to make intelligent decisions, we need to properly represent the game state. In our research, the game state consists of the following key elements:

Player's hand: For each player, we keep track of the cards they currently hold. Each card is composed of a color and a number. For example, "red 2" represents a red card with the number 2.

Discard pile: We keep track of the cards in the discard pile to make judgments based on the previous card. The cards in the discard pile are arranged in the order they were played.

Deck: The deck holds the remaining cards from which players can draw.

Current player: We record which player's turn it is to play.

By combining these key elements, we can fully represent the game state. For example, a typical game state might look like this:

Player A's hand: red 0, blue 1, green 3

Player B's hand: blue 0, yellow 2, green 4

Discard pile: red 2, blue 2, green 2

Remaining deck: red 1, red 3, blue 3, yellow 0, yellow 1, yellow 3, yellow 4, green 1, green 2, green 3, green 0

Current player: 1

By continuously updating and maintaining these elements in real-time, AI agents can make card-playing decisions based on the current state and select the optimal strategy for playing cards. Our goal is to enable AI agents to make intelligent decisions in each round based on the game state and rules, in order to achieve victory as quickly as possible.

2.2. The Basic Principle of Mini-Max Algorithm

When solving the card game's card decision problem, the Mini-Max algorithm is widely used in game tree search. This classic algorithm allows the AI agent to predict the consequences of different decisions within a limited search depth and choose the optimal decision strategy. In this section, we will introduce the basic principles of the Mini-Max algorithm and its application in "CardCraft Champ". We will also explain the evaluation function we have adopted and its impact on card decision-making.

2.2.1. The Brief Introduction of Mini-Max Algorithm. The Mini-Max algorithm is a game tree search method aimed at finding the optimal decision strategy that maximizes our gain in the worst-case scenario for the opponent [5]. It simulates the decision-making process between players by constructing a game tree. At each level of the tree, when it is a player's turn to make a decision, that player will choose the branch that maximizes their gain, while the opponent player will choose the branch that minimizes their gain, thus forming the concept of "maximizing the minimum value".

The core idea of the Mini-Max algorithm is to recursively search the game tree and compute the score for each decision branch. For player A, they will choose the maximum score in each branch; for player B (the opponent player), they will choose the minimum score in each branch. After searching the entire game tree, player A will choose the branch with the highest score as the best decision, while player B will choose the branch with the lowest score as the best countermeasure.

DOI: 10.54254/2755-2721/73/20240390

2.2.2. Design and Application of Evaluation Function. In "CardCraft Champ", our evaluation function is based on the difference in the number of cards between players. Specifically, we use the evaluation function of "opponent's card count minus our card count". The larger this value is, the fewer cards the opponent has, and the more advantage our AI has. Therefore, during the search process of the Mini-Max algorithm, the AI agent will try to select branches with larger evaluation function values to maximize our advantage.

The design of the evaluation function plays a crucial role in determining the decision-making ability of the AI agent. In "CardCraft Champ", we measure the favorability of the current situation for the AI agent by the value of the evaluation function. When the search reaches the game end state (i.e., one player wins), the value of the evaluation function will be positive infinity or negative infinity, depending on whether we win or the opponent wins. In non-terminal states, the value of the evaluation function fluctuates between positive and negative infinity, reflecting the current situation's quality.

2.2.3. The Application of Mini-Max in "CardCraft Champ". In "CardCraft Champ", we combine the Mini-Max algorithm with the game rules and state representation to obtain the optimal card-playing decision within a limited search depth. During each player's turn, the AI agent will search the game tree, explore different card-playing strategies, and calculate the evaluation function value for each branch. Then, the AI agent will choose the branch with the maximum evaluation function value as the best card-playing strategy in order to achieve the most favorable situation.

However, due to the potentially large search space of the game tree, a complete search may consume a significant amount of time and computational resources. Therefore, we usually set a finite search depth. For example, we set the search space of the game tree to three levels and combined it with the Alpha-Beta pruning technique to reduce unnecessary searches. In this way, "CardCraft Champ" can make decisions within a reasonable amount of time while ensuring the quality and intelligence of the card-playing strategy.

2.3. Optimization Strategy of Alpha-Beta Pruning

Alpha-Beta pruning is a technique used to reduce the number of branches searched in a game tree, in order to exclude branches that are unlikely to lead to a better solution and improve search efficiency. This technique relies on two parameters: Alpha and Beta. During the search process, if the score of a node exceeds the Alpha-Beta range, it can be pruned because the score of that node is irrelevant to the current player.

Specifically, for player A (the side maximizing the score), if the score of a node is greater than or equal to the Beta value, the subsequent branches of that node do not need to be further searched, because player B (the side minimizing the score) will not choose that node, and player A has already found a better branch. Similarly, for player B, if the score of a node is less than or equal to the Alpha value, the subsequent branches of that node also do not need to be searched, because player A will not choose that node.

In "CardCraft Champ," we combine the Alpha-Beta pruning technique with the Mini-Max algorithm to improve the efficiency of the AI agent's card-playing decisions in the game. Specifically, during the game tree search, the AI agent will decide whether to prune a node based on the node's score compared to Alpha and Beta. This approach effectively reduces unnecessary searches and improves search speed.

In the optimization strategy of Alpha-Beta pruning, we also need to pay attention to the order of search. Generally, we will first search branches with higher evaluation function values (i.e., larger Alpha values) in order to discover more promising decisions earlier. When searching the branches of the opponent player, we will search branches with lower evaluation function values (i.e., smaller Beta values) first to find the opponent's possible optimal countermove.

By introducing the optimization strategy of Alpha-Beta pruning, we are able to significantly improve the search speed of "CardCraft Champ" while maintaining the quality of decisions. However, it is important to note that the effectiveness of Alpha-Beta pruning depends on the search order and the design of the evaluation function. Improper search order may result in premature pruning and missing

DOI: 10.54254/2755-2721/73/20240390

out on better solutions, while unreasonable design of the evaluation function may lead to excessive search branches and reduce the effectiveness of pruning.

3. Analysis of the Experimental Results

In our study, we evaluated the performance and effectiveness of "CardCraft Champ" in the UNO card game through experiments. The initial version of "CardCraft Champ" was implemented using Python, and we verified the accuracy and intelligence of its card-playing decisions through extensive testing and competition.

Our experiments aimed to validate the performance of "CardCraft Champ" in the UNO game, particularly its accuracy and intelligence in making card-playing decisions. To do this, we designed a series of adversarial experiments, where "CardCraft Champ" played against random players or other card-playing strategies to evaluate its win rate and average number of game rounds.

We implemented the algorithm of "CardCraft Champ" using the Python programming language and created an environment to simulate the UNO game. In each experiment, we had "CardCraft Champ" play against different opponents for a certain number of matches and recorded its win rate, average number of game rounds, and performance in card-playing decisions. As the initial version, our focus was on validating the basic functionality of the algorithm, while future versions will further optimize and improve its performance.

Through experimental testing, we obtained preliminary performance results for "CardCraft Champ". When playing against random players, "CardCraft Champ" was able to achieve a win rate of over 50%, which is an encouraging result. Although this is only the initial version, it demonstrates the card-playing decision-making ability of "CardCraft Champ" in the UNO game.

We also observed that "CardCraft Champ" typically won within a shorter number of game rounds when playing against random players. This indicates that "CardCraft Champ" is able to adapt to the game situation and quickly find opportunities for victory in its card-playing decisions. However, compared to other more complex card-playing strategies, there may still be room for improvement for "CardCraft Champ" in certain specific situations.

4. Conclusions

Building upon the initial implementation and testing of the "CardCraft Champ 1.0" version, we have come to realize that this is just the beginning, and future research and improvements will make "CardCraft Champ" a more intelligent and powerful UNO game AI agent.

4.1. Algorithm Optimization and Expansion

Although "CardCraft Champ 1.0" has already demonstrated promising initial results, we believe there is still significant room for algorithm optimization and expansion. We will strive to introduce more advanced game tree search algorithms, such as Monte Carlo Tree Search (MCTS), along with methods that combine deep reinforcement learning. These algorithms can make more intelligent decisions in more complex game scenarios, allowing "CardCraft Champ" to adapt to different opponents and strategies.

4.2. Adaptive Decision Model

In order to better deal with different opponents and changing game situations, we plan to study adaptive decision-making patterns. By analyzing opponents' playing strategies and historical information, "CardCraft Champ" can adjust its own decisions based on different situations to achieve a higher win rate. This requires the introduction of more complex decision models and opponent modeling methods, but it will make "CardCraft Champ" more competitive in diverse matches.

4.3. Integration of Deep Reinforcement Learning

In future upgrades, we will consider integrating deep reinforcement learning (DRL) with the design of "CardCraft Champ". DRL enables AI agents to learn more complex strategies and tactics through

DOI: 10.54254/2755-2721/73/20240390

interactions with the environment. Using a neural network-based approach, "CardCraft Champ" can learn superior card-playing strategies from a large number of matches, thereby performing better in actual battles.

4.4. Enhanced State Representation and Feature Extraction

The performance of "CardCraft Champ" largely depends on the quality of the state representation. We will continue to research how to accurately represent the game state and develop more efficient feature extraction methods. This will help improve "CardCraft Champ's" understanding and decision-making capabilities regarding the game state, enabling it to better adapt to different situations.

4.5. Expansion of Multi-agent Environment

Although the current version primarily focuses on a single intelligent agent for the UNO game, we hope to expand "CardCraft Champ" into a multi-agent environment in the future, such as a game with multiple opponents to compete against. This will provide greater challenges for "CardCraft Champ" while also offering a new perspective for researching multi-agent gaming.

4.6. Interact and Learn with Human Players

In addition to playing against AI agents and random players, we also hope that "CardCraft Champ" can interact with human players. By competing against human players, "CardCraft Champ" can learn from their strategies and ways of thinking, further enhancing its intelligence level.

"CardCraft Champ" as the first version of the card game AI agent has achieved some satisfactory results. However, we are well aware that this is just a starting point, and future research and improvements will continuously evolve "CardCraft Champ" into a more intelligent and superior AI agent. Through algorithm optimization, adaptive decision-making patterns, deep reinforcement learning, better state representation, and expanding into multi-agent environments, we are confident that "CardCraft Champ" will continue to develop in future research, bringing new breakthroughs to the field of card game research and practice.

References

[1] Demaine, E. D., Demaine, M. L., Uehara, R., Uno, T., & Uno, Y. (2010). UNO is hard, even for a single player. In Springer eBooks (pp. 133–144). https://doi.org/10.1007/978-3-642-13122-6_15

[2] Zha, D., Xie, J., Ma, W., Zhang, S., Lian, X., Hu, X., & Liu, J. (2021). DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning. ICML.

[3] Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.

[4] Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Li, Z., Qin, T., Liu, T., & Hon, H. (2020). Suphx: Mastering Mahjong with Deep Reinforcement Learning. arXiv (Cornell University). https://arxiv.org/pdf/2003.13590

[5] Rivest, R. L. (1987). Game tree searching by min/max approximation. Artificial Intelligence, 34(1), 77–96. https://doi.org/10.1016/0004-3702(87)90004-x