Purble Pairs: 2 Player Edition

An AI-Based Memory Matching Game

Artificial Intelligence (Course Project)

Dinesh Kumaran K (CS23B2057)   Shirish Giroti (CS23B2041)

April 8, 2025

Abstract

This report presents the design, implementation, and AI strategies used in building the game **Purble Pairs: 2 Player Edition**, a two-player memory card matching game that incorporates advanced features like penalty reshuffles, match ownership, and an intelligent adversarial AI player. The project explores how a seemingly simple matching game can become computationally challenging under specific constraints, and discusses how Adversarial Search and memory-based strategies were employed to create a rational AI. We also analyze the complexity of the problem and justify why it qualifies as NP-Hard.

1 Introduction

Memory card games challenge players to remember and match hidden pairs of cards based on their positions. We designed an AI-powered version of this game, extending it to a two-player format where one player is a human and the other is an AI agent. The game is interactive, competitive, and built entirely in Python.

2 Game Description

The game consists of a 4x4 board containing 8 distinct fruit emojis, each appearing twice. Players take turns flipping two hidden cards. If a match is found, they score a point and retain their turn. If not, the cards are flipped back, and the turn passes to the other player.

Key Features:

- Turn-based interaction (AI vs Human)
- Score tracking and ownership of matched pairs
- Penalty reshuffling mechanism after 3 consecutive misses
- Memory-based AI with partial observability

3 Game Rules

1. The board is initialized with pairs of random fruit emojis.
2. Each player flips two cards on their turn.
3. If the cards match, they are removed from play and assigned to the player.
4. A player retains their turn on a match.
5. If a player makes 3 consecutive unsuccessful attempts, a penalty reshuffle occurs:
   - One previously matched pair (if available) is returned to the board.
   - All unmatched cards are reshuffled.
   - Both players' visible and revealed states are reset.
6. The game ends when all pairs are matched.

4 Constraints

We introduced several constraints to increase the complexity of the problem:

- **Visibility Constraint:** Cards are only visible when flipped or matched.
- **Memory Constraint:** AI only remembers revealed cards.
- **Ownership Constraint:** Each pair is owned by a player.
- **Penalty Rule:** Reshuffling is triggered after 3 consecutive mismatches by the same player.
- **Turn Retention:** Players keep their turn if they successfully match a pair.

5 AI Techniques Used

5.1 Adversarial Search

Our AI agent uses a form of **Adversarial Search**, similar to the **Minimax algorithm**. This technique is commonly used in two-player zero-sum games like Chess or Tic-Tac-Toe.

Key Properties

- The AI models the game as a decision tree.
- Each state represents a board configuration.
- The AI chooses actions that maximize its expected outcome while minimizing the player's.
- In this project, the AI uses its memory to simulate smart matching behavior.

While we did not implement a full Minimax tree due to the high branching factor and hidden information, our AI mimics rational adversarial reasoning through memory-based heuristics and optimal action selection when possible.

5.2 AI Memory and Strategy

- AI stores all card positions that have been revealed.
- It prioritizes matching known pairs from memory.
- If no known match exists, it selects two unseen cards randomly.
- AI avoids already matched cards.
- If a known match is available, it takes the guaranteed move to increase score.

6 NP-Hardness Justification

Although the basic card-matching game is polynomial in nature, our version introduces elements that make optimal play computationally intensive:

- **Hidden Information:** The AI only sees partial board state.
- **Reshuffling:** The game board changes dynamically, making prior knowledge partially invalid.
- **Constraint Interaction:** Scoring, turn retention, and penalty reshuffles introduce dependencies.
- **Optimization Objective:** Finding the best sequence of moves to win becomes a combinatorial search problem.

With these elements combined, the game resembles Constraint Optimization with incomplete information, placing it in the category of NP-Hard problems due to exponential growth in possibilities and interleaving constraints.

7 Optimizations

To ensure smooth gameplay and effective AI:

- We store AI memory in a hash map to allow constant-time lookups.
- Only unmatched cards are reshuffled to preserve fairness.
- Previously matched cards are not revealed again.
- Penalty reshuffle excludes opponent's matched pairs to maintain competitiveness.

8 Code Snippets and Explanation

AI Memory Matching Strategy

Listing 1: AI uses memory to find matching pairs

```
for (r1, c1), fruit1 in self.ai_memory.items():
    if self.matched[r1][c1]:
        continue
    for (r2, c2), fruit2 in self.ai_memory.items():
        if (r1, c1) != (r2, c2) and fruit1 == fruit2 and not self.matched[
            ↪ r2][c2]:
            return (r1, c1), (r2, c2)
```

Explanation: The AI agent loops through its memory and tries to find two cards with the same fruit. If such a match is found and the cards are still unmatched, the AI selects them.

Penalty Reshuffling Logic

Listing 2: Penalty reshuffle after 3 misses

```
if self.consecutive_misses[self.current_player] == 3:
    self.penalty_reshuffle()
    self.consecutive_misses = [0, 0]
```

Explanation: This rule adds strategic pressure. After 3 consecutive misses by a player, a previously matched pair is returned and the board is reshuffled, increasing uncertainty.

Tracking Card Ownership

Listing 3: Matched cards assigned to player

```
self.matched[row1][col1] = True
self.matched[row2][col2] = True
self.owners[row1][col1] = self.owners[row2][col2] = self.current_player
self.scores[self.current_player] += 1
```

Explanation: This ensures every matched pair is attributed to the correct player, allowing for accurate scoring and pair reversion logic during penalties.

AI Memory Update on Reveal

Listing 4: AI records any revealed card

```
self.revealed[row][col] = self.board[row][col]
self.ai_memory[(row, col)] = self.board[row][col]
```

Explanation: Whenever a card is revealed during play (either by AI or human), the AI updates its internal memory with that card's position and symbol.

9 Conclusion

The game Purble Pairs: 2 Player Edition combines the fun of memory games with the challenge of intelligent gameplay. We successfully implemented an AI opponent using principles of adversarial reasoning and memory-based learning. The interaction of game rules, constraints, and partial observability results in a problem that exhibits NP-Hard characteristics. Our work demonstrates how classical AI techniques can be used to create competitive, dynamic, and engaging gameplay experiences.

References

- Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach.*
- Game design theory on NP-completeness and constraint-based logic.