Report: Diamonds Game

Akanksha Narula

Introduction

The goal of this task is to converse with GenAI(ChatGPT GPT3 .5 version) in order to acquaint it with the Diamonds game, which it is unfamiliar with. The assignment's primary goal is to make sure that the game's rules and intricacies were thoroughly understood by GenAI. The next step is to have GenAI come up with winning strategies after it has a firm grasp of the game. The computer programme will then use these methods to let GenAI play Diamonds versus a human player. This assignment's main objective is to evaluate GenAI's comprehension, planning, and performance in a challenging, strategy-based game.

Teaching genAI the Game

First, GenAI was told about the rules of the Diamonds game that are:
• The game is played among 2-3 players, consisting of 13 rounds, each dealt a suit of cards excluding the diamond suit.
• The diamond cards are shuffled, placed face down, and auctioned off one by one.
• When a diamond card is revealed, all players bid for it by placing one of their own cards face down. The point system for cards is: 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A in increasing order.
• After all players have chosen their bid card, the cards are revealed, and the diamond card is awarded to the player with the highest bid in points. The player who wins the bid receives the points of the diamond card added to their score.
• If multiple players tie for the highest bid with the same card, the points from the diamond card are divided equally among them.
• The player with the most points at the end of all rounds wins the game.

In order to improve the AI's gameplay performance, the strategies explored for the Diamond Cards game progressed from fundamental ideas to more complex techniques. The AI was initially configured to bid in accordance with the diamond card's value and
the current round number, placing the lowest bid for low point cards and the highest bid for high point diamond cards. Then, a sophisticated tactic was put forth, which included components like card management, assessing the cards of opponents,
Strategies for the end game and risk-reward balance. In order to make better-informed bids, the AI's decision-making process was further improved by taking into account the cards played in earlier rounds. Challenges continued to routinely outperform a human player in spite of these improvements. Additional changes were proposed, like a probabilistic approach based on the mean value of remaining cards.

Iterating upon Strategy

Firstly, I asked GenAI to play the game with me to ensure we both and the same and correct understanding of the game.
Then, I asked it to correct the errors in its understanding.
Later, I discussed some optimum strategies to maximize my chances of winning.
The plan called for prudent card management, estimation of opponents' cards, risk-reward analysis, and well-informed decision-making during the last rounds.

Then, GenAI was instructed to write a Python programme to carry out this tactic. The AI player in the programme uses the previously mentioned approach to determine its bids in a simulation of the game of diamond cards. When the software was being created, the
Techniques were constantly enhanced and improved. More complex tactics were added as a result of further cues, such taking opponent actions and played cards into account. This iterative approach to improvement and improvisation
with the intention of improving the program's strategies' efficiency. Link to the conversation: https://chat.openai.com/share/777b2eb1-cb9e-48c8-a7f3-1a30cb3d994b

Analysis and Conclusion

After developing the AI strategy for the Diamonds game, let's analyze its effectiveness and draw conclusions. The AI strategy is designed to make bidding decisions based on several factors, including the cards it holds, the value of the diamond card, and an estimation of the opponent's cards. Here's a breakdown of how the strategy works:

- **Estimating Opponents' Cards:** The AI estimates the opponent's cards based on the cards they've won in previous rounds.
- **Average Value Calculation:** It calculates the average value of its remaining cards to determine whether to bid high or low.
- **Bid Decision Making:** Based on the comparison between the diamond card's value and the average value of its cards, the AI decides whether to bid high or low. Additionally, it considers the likelihood of the opponent running out of high cards.

In conclusion, the AI strategy demonstrates a strategic approach to bidding in the Diamonds game. By analyzing its own cards, estimating opponent cards, and adapting bidding strategies based on the value of the diamond card, the AI aims to maximize its chances of winning rounds and accumulating points.
To further illustrate the implementation of the AI strategy, here's the Python code snippet:

```
import random

cards = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, 'T': 10, 'J': 11, 'Q': 1
diamond_cards = list(cards.keys())
random.shuffle(diamond_cards)

def play_round(player_card, ai_card, diamond_card):
    print(f"Player bid: {player_card}, AI bid: {ai_card}")
    if cards[player_card] > cards[ai_card]:
        return 'Player'
    elif cards[player_card] < cards[ai_card]:
        return 'AI'
    else:
        return 'Tie'

def ai_strategy(ai_cards, round, diamond_card, played_cards):
    estimated_opponent_cards = [card for card in cards.keys() if card not in ai_cards and card no
    avg_ai_card_value = sum(cards[card] for card in ai_cards) / len(ai_cards) if ai_cards else 0

    if cards[diamond_card] > avg_ai_card_value and len(ai_cards) > 0:
        return ai_cards.pop(ai_cards.index(max(ai_cards, key=cards.get)))
    elif cards[diamond_card] <= avg_ai_card_value and len(ai_cards) > 0:
        return ai_cards.pop(ai_cards.index(min(ai_cards, key=cards.get)))
    elif estimated_opponent_cards and max(estimated_opponent_cards, key=cards.get) < max(ai_cards
        return ai_cards.pop(ai_cards.index(max(ai_cards, key=cards.get)))
    else:
        return ai_cards.pop() if ai_cards else None

def game():
    player_score = 0
    ai_score = 0
    player_cards = list(cards.keys())
    ai_cards = list(cards.keys())
    random.shuffle(player_cards)
    random.shuffle(ai_cards)
    played_cards = []

    for round in range(13):
        diamond_card = diamond_cards.pop()
        print(f"Diamond card: {diamond_card}")
        player_card = input("Enter your bid: ")
        while player_card not in player_cards:
            player_card = input("Invalid card. Please enter a valid card from your hand: ")
        player_cards.remove(player_card)
        played_cards.append(player_card)
        ai_card = ai_strategy(ai_cards, round, diamond_card, played_cards)
        if ai_card:
            played_cards.append(ai_card)
        winner = play_round(player_card, ai_card, diamond_card)
        if winner == 'Player':
            player_score += cards[diamond_card]
            print("You win this round!")
        elif winner == 'AI':
            ai_score += cards[diamond_card]
            print("AI wins this round!")
        else:
            player_score += cards[diamond_card] / 2
            ai_score += cards[diamond_card] / 2
            print("It's a tie!")
        print(f"Scores -> Player: {player_score}, AI: {ai_score}\n")

    if player_score > ai_score:
        print("You win the game!")
    elif player_score < ai_score:
        print("AI wins the game!")
    else:
        print("It's a tie!")

game()
```

Through the integration of such a strategy and the continuous refinement of bidding techniques, players and AI systems can enhance their gameplay experience and competitiveness in the Diamonds game.