Arda Küçük Group11 Assignment report

This report created for describing and explaining the structure of AwesomeNegotiator class designed for 2024 ANAC League. Awesome negotiator is an AI agent designed to manage the negotiation process continuously analyzing the opponent's behaviors and adjusting its strategy based on this information. At every step of the negotiation process the agent's decision-making process is towards to predicting potential acceptable offer from the opponent and determining the best response to these offers. The agent aims to maximize its own benefits while also contributing positively to the negotiation process.

Class Structure and Functions

- On_preferences_changed:

  This Method is called when there are changes in the agent's preferences. It calculates all possible outcomes within the domain and stores it in self.rational_outcomes. Additionally, it sets the opponent's reservation value as a starting point equal to its own reservation value.

- Call:

  Determines the agent's response based on the current negotiation. If the offer from the opponent is not acceptable a new offer is generated through the bidding_strategy method

- Acceptance_strategy:

  This function decides whether the current offer should be accepted or not. It compares the utility of the current offer against a adjusted threshold based on time progression. If the offer exceeds this threshold or close to the highest utility observed from the opponent as the deadline approaches, it is accepted.

- Bidding_strategy

  Determines the counteroffer to be presented to the opponent. As time progresses in the negotiation the strategy evaluates offers considering both the agent's and the opponent's benefits to maximize overall utility.

- Update_partner_reserved_value

  Updates the estimated reservation value of the opponent based on the most recent offer received. If the utility from the opponent's last offer is lower than what was previously stored it indicates as a possible lower boundary which updates the reserved value.

Acceptance Strategy

The acceptance strategy function is a crucial element of the agent's negotiation process. This function is specifically designed to decide whether to accept the offer currently on the table based on a dynamic assessment of the negotiation context.

Key components

- 1. Dynamic Threshold Based on Time Progression
  The function calculates a threshold that decreases as the negotiation progresses. This is based on the formula dynamic_threshold = (2 - 1.5 * time_elapsed) * self.ufun.reserved_value, which adjusts the acceptance threshold lower as more time elapses. This dynamic thresholding reflects the urgency and flexibility needed as the negotiation approaches its deadline. Influanced by "Accepting optimally in automated negotiation with incomplete information".

- 2. Decision Based on Utility Comparison:
  The function checks if the utility of the current offer exceeds the dynamically calculated threshold. If the offer's utility surpasses this threshold, it is immediately accepted. Influanced by Effective acceptance conditions in real-time automated negotiation," Decision Support Systems.

- 3. Endgame Strategy Enhancement:
  As the deadline nears (remaining_time < 0.2), the strategy includes a check against the highest utility offer observed so far within the negotiation (‘max_past_utility’). If the current offer's utility is at least 90% of this maximum value, it is accepted. This part of the strategy aims to secure a favorable outcome before the negotiation window closes, minimizing the risk of leaving the negotiation table with a suboptimal deal or no deal at all. Influanced by “Compromising strategy based on estimated maximum utility for automated negotiating agents”.

  After the first implementation of this code we can see that our score is highly increased I aimed to get a better score then conceder to start. Here is my agent’s score after implementing acceptance strategy:

```
Will run 4 negotiations on 1 scenarios between 2 competitors
negotiations ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:07
            strategy     score
0           Conceder  0.797052
1  AwesomeNegotiator  0.688255
```

Bidding Strategy

Key Components

- 1. Trade off Strategy

  The strategy uses a time-dependent weighting system where the agent's priority shifts from maximizing its own utility early in the negotiation to considering the opponent's utility more as the deadline approaches. The weights adjust dynamically as my_weight = 1 - time_elapsed and their_weight = time_elapsed, reflecting a strategic shift towards compromise over time. Influanced by "Trade-off Strategy" from negotiation literature, where negotiators balance multiple decision variables to increase the overall welfare of all parties involved. This approach helps in reaching agreements that are acceptable to both parties by making trade-offs between different negotiation issues.

- 2. Concession Strategy

  If no offers exceed the combined utility of the previous best offer, the strategy may still choose an offer close to this best utility as a concession. This is part of ensuring that the negotiation moves forward and reflects a willingness to compromise, particularly as the negotiation nears its conclusion.

  I influenced by A negotiation meta strategy combining trade-off and concession moves and using similarity criteria to make issue trade-offs in automated negotiations. Bidding strategy that involves combining different negotiation tactics, specifically trade-offs and concessions. This influence is reflected in the strategy's ability to dynamically adjust offers based on a combination of the agent's and the opponent's utilities, emphasizing a balanced approach to advancing the negotiation towards a mutually acceptable agreement.

Reserved Value

- 1. Utility Function Utilization:

This method evaluates the utility of the latest offer made by the opponent using the opponent's utility function (opponent_ufun). If the utility derived from this offer is lower than the current stored reservation value (partner_reserved_value), the reservation value is updated to this new lower value. This adjustment reflects the lowest utility the opponent might accept based on the current negotiation context. Updating the reservation value based on this new information allows your agent to adapt its negotiation strategy

accordingly, aiming to secure a more advantageous position by understanding the lowest utility the opponent is potentially willing to settle for.

- 2. Updating Offer Strategies:

Following the update of the reservation value, the list of rational outcomes (rational_outcomes) is also updated to reflect this new reservation threshold. This update filters out any outcomes from the rational outcomes list that now fall below the newly adjusted reservation value. By doing so, the agent ensures that all future offers it considers are above or equal to the updated reservation value, which can enhance the effectiveness of the offers it presents in later stages of the negotiation. This strategy helps maintain a high standard for the offers and aligns the negotiation actions with the most current understanding of the opponent's negotiation threshold, potentially leading to more successful negotiation outcomes.

Quantifying the agent's performance

1.Main Domain

I tested main domain for 10 times so I can see my percentage in this domain.

Test1: 0 AwesomeNegotiator 0.275779 -1 Conceder 0.275490 8s
Test2: 0 Conceder 0.228300 -1 AwesomeNegotiator 0.096782 8s
Test3: 0 Conceder 0.403729 -1 AwesomeNegotiator 0.310993 10s
Test4: 0 Conceder 0.568815- 1 AwesomeNegotiator 0.324796 7s
Test5: 0 Conceder 0.330282 -1 AwesomeNegotiator 0.289102 7s
Test6: 0 Conceder 0.515639 -1 AwesomeNegotiator 0.275266 8s
Test7: 0 Conceder 0.673616 -1 AwesomeNegotiator 0.541514 8s
Test8: 0 Conceder 0.464263 -1 AwesomeNegotiator 0.383862 9s
Test9: 0 Conceder 0.306957 -1 AwesomeNegotiator 0.153313 9s
Test10: 0 Conceder 0.525583 -1 AwesomeNegotiator 0.369664 9s