CAI Group 59 - Agent Negotiation

Andrei Boboc , Jyotiradityaa Jaiman , Mara Mih , Zeryab Alam

1 Introduction

CSE3210 Assignment 3 focused on implementing a negotiation mechanism between artificial agents. Two agents would be involved in each round of negotiation, during which they adopted the turn-based protocol for reaching an agreement. Taking turns proposing bids of different utilities for the agents, the idea is for both parties to agree on the most beneficial outcome, considering relevant literature on the objectively best possible result. Specifically, we are building towards an outcome close to points on the Pareto Optimal Frontier of negotiation between agents, defined, for instance, in the book by Raiffa (1982).

Our group's approach is to set high standards off the bat and adapt the Kalai-Smordinsky solution - taking into consideration equal utility gains for both agents - by weighing bids based on their distance to our best bid. As time runs down, we lower our expectations in order to reach an agreement, an important focus in our eyes, but still keep a minimum utility level - the reservation value - we would like to stay above. All of these choices, as will be shown in this document, are based on an interest in building an ambitious agent with high results in both its own utility and its opponents' utilities, or at least one of those if the negotiation takes too long. We believe that our experimental results prove our solution to be suitable.

2 Agent Description: AgentQT

Our mechanism implements an ambitious agent whose strategy relies on one major trade-off: either our opponents offer us a bid that, together with our utility value, ensures a highly successful outcome (high Nash product and social welfare), or at least one of us reaches high utility until time runs out. Our bidding strategy and acceptance conditions are further detailed below, but in essence, we keep our opponent's best recent bid in mind and use well-defined predictions in order to compute counter offers. Furthermore, we weigh bids in relation to their "distance" to the best bid, thus removing any randomness from the negotiation process. The agent does implement a mechanism where time matters, too, as we wish to reach an agreement as long as it is not too harsh (below our reservation value, comparable to realistic expectations based on relevant referenced literature).

3 Bidding Strategy

Our negotiating agent employs a multifaceted and adaptive bidding strategy designed to optimize outcomes while navigating the negotiation process with its counterpart. This strategy encompasses several crucial elements: the initial offer, handling the opponent's offer and the counter offer.

3.1 Initial Offer

The initial offer is pivotal in determining the opening bid our negotiating agent presents to its counterpart. It begins by extracting the negotiation domain from our agent's profile and generating a comprehensive list of all possible bids within that domain. It then iterates through each potential bid, calculating the utility of each bid based on our agent's preferences. These bid-utility pairs are stored, allowing the function to identify the bid with the highest utility, which becomes our agent's initial offer. By sorting the bids based on utility, our agent ensures it starts the negotiation from a position that maximizes the likelihood of achieving a favorable outcome, setting the stage for productive negotiations with its counterpart.

Furthermore, this initial offer is stored in memory for future reference, aiding the agent in subsequent decision-making during the negotiation process.

3.2 Dealing With Opponent Offers

Upon receiving an offer from the opponent, the agent first assesses whether it meets their reservation value of 0.35. If it falls short, the agent reiterates their best bid, prompting the opponent to improve their offer. Concurrently, the agent keeps track of the frequency of such occurrences. If the opponent persists with hardline stances, the agent gradually increases their threshold for conceding, ensuring that concessions only follow significant deviations from their desired terms.

If the offer surpasses their reservation value, the agent evaluates whether it aligns with their acceptance criteria. Should it meet their conditions, the agent promptly extends an acceptance offer to finalize the agreement. If not, the agent engages in negotiation by crafting a counter-proposal and subsequently presenting it to the opponent.

```
# Check if there's a previous bid from the opponent
if self.last_received_bid:
    # The offer should firstly be better than reservation bid
    if self.check_reservation() or self.hardline:
        if self.hardline:
            self.hardline = False
        # Check if the opponent's bid is acceptable and accept
        if self.accept_condition(self.last_received_bid):
            action = Accept(self.me, self.last_received_bid)

        # If not acceptable, make a counter-offer
        else:
            bid = self.find_bid()
            self.current_offer = bid
            action = Offer(self.me, bid)

    # Opponent offer is below reservation value, send the best bid to force a better offer
    else:
        if self.counter >= self.conceding_factor:
            self.hardline = True
            self.counter = 0
            self.conceding_factor += 5
        else:
            self.counter += 1
        action = Offer(self.me, self.best_bid)
```

Figure 1: Handling Opponent Offers

3.3 Counter Offer

To formulate a counteroffer, our agent adapts the Trade-Off Strategy (P. Faratin (2002)), leveraging opponent model utility values to optimize negotiation outcomes. Here's an elaboration on how our agent integrates this strategy:

Our agent initiates the counteroffer process by analyzing the last 10 bids received from the opponent, identifying the bid with the highest utility for the agent. This bid serves as a reference point for determining the agent's best value.

Using a sorted list of all bids, our agent then calculates two crucial values: the opponent value and the agent value. The opponent value represents 30% predicted utility from the opponent's perspective and 70% utility for our agent from the opponent's offer, incorporating insights from the opponent model. Conversely, the agent value considers 30% predicted utility for the opponent and 70% utility for our agent from our last offer.

```
opponent_best_offer = max(self.opponent_last_10_values, key=lambda bid: float(self.profile.getUtility(bid)))

opponent_offer_our_util = float(self.profile.getUtility(opponent_best_offer))
opponent_offer_their_util = float(self.opponent_model.get_predicted_utility(opponent_best_offer))
our_offer_our_util = float(self.profile.getUtility(self.current_offer))
our_offer_their_util = float(self.opponent_model.get_predicted_utility(self.current_offer))

opponent_best_value = (0.3 * opponent_offer_their_util) + (0.7 * opponent_offer_our_util)
agent_value = (0.3 * our_offer_their_util) + (0.7 * our_offer_our_util)

best_value = float(self.profile.getUtility(self.best_bid))
```

Figure 2: Counter Offer Key Values

Next, these values are averaged to establish a range from the opponent's perceived value to our agent's best value, typically 1, but possibly adjusted if concessions are made. This concession is made by decreasing the agent's best value by 0.2 only if the opponent has not been hard-lining by making offers constantly better than our agent's reservation value. Bids falling within this range are considered, and additional weight is assigned to favor bids slightly more beneficial to our agent.

Finally, a random bid meeting these criteria is selected as the counteroffer. This comprehensive approach ensures that the counteroffer strikes a delicate balance, offering concessions to the opponent while advancing our agent's objectives. By incorporating opponent model values and strategically navigating trade-offs, our agent fosters a negotiation environment conducive to achieving mutually beneficial outcomes.

```
average_value = (opponent_best_value + agent_value)/2

filtered_bids = [(bid, value) for bid, value in self._allBids if average_value <= value <= best_value]
print("filtered list size")
print(len(filtered_bids))
if filtered_bids:
    # Calculate weights based on the distance from the best bid
    weights = [1 / (abs(float(best_value) - float(value)) + 0.1) for bid, value in filtered_bids]

    # Normalize weights to sum up to 1
    total_weight = sum(weights)
    normalized_weights = [weight / total_weight for weight in weights]

    # Randomly select one bid from the filtered list with weighted probability
    selected_bid_index = random.choices(range(len(filtered_bids)), weights=normalized_weights)[0]
    selected_bid = filtered_bids[selected_bid_index]

    # Return the bid
    return selected_bid[0]
else:
    return self.current_offer
```

Figure 3: Counter Offer Finalization

4 Acceptance Strategy

The agent has four acceptance conditions, two of which must be satisfied in order for a bid to be accepted. A separation is made between the "early game" of the round - first 7 seconds - and the late game, in which we slowly become more lenient in order to reach an agreement - still forcing an outcome of utility above our reservation value.

The first condition is the one on time: as time goes on, we are compromising more. More specifically, after half the round time plus one second, we start a sequence in which, with each second, we accept bids if their utility is above a certain value: from 0.8 down to 0.5 in the last second. These are still quite high values based on two relevant references we can mention: firstly, all groups' experimental results, and secondly, the referenced paper by Catholijn M. Jonker, Dmytro Tykhonov and Koen Hindriks (2011). These sources show that utility values in the range of 0.5-0.8 for our agent are above average and thus desirable. Since we are running out of time, regardless of our opponent's utility, we have to accept values as close to the Pareto Optimal Frontier as possible.

```
    # Progression based acceptance strategy
    strategy1 = ((progress > 0.97
                  or (progress > 0.9 and received_bid_utility > 0.5)
                  or (progress > 0.8 and received_bid_utility > 0.6)
                  or (progress > 0.7 and received_bid_utility > 0.7)
                  or (progress > 0.6 and received_bid_utility > 0.8))
                 and (received_bid_utility >= reservation))
```

Figure 4: Code for condition 1

Naturally, we do want to maximize our Nash product too, thus not seeking a maximum value for ourselves at the expense of minimizing the opponent's utility totally. Having these expectations quite late into the negotiation shows our agent's ambition and potential for high success until the very

end - it is true that we do not focus on quickly getting an agreement. After 0.97 of the total round time, we are forced to accept it as long as the bid is greater than or equal to our reservation value. Figure 4 illustrates the implementation of this condition.

In addition to guaranteeing that the negotiation has some sort of desirable outcome through ensuring that the reservation value is satisfied, the time-based condition makes our agent remain ambitious for a significant amount of time. During the first 6 seconds of the round, we hope for strategy number four to apply, which favours great results in terms of Nash product, Social welfare, and both agents' utilities. If these highly desirable outcomes do not happen, we finally lower our expectations slowly but surely and eventually agree to lower values down to our reservation value. This is the essence of condition number one.

Condition number two, which we call the prediction-based acceptance strategy, also only comes into effect after 7 seconds, a quite significant portion of the round, thus allowing plenty of time for better results. Nevertheless, after this amount of time, we start predicting the next bid that we would offer via our find bid method that generates the next bid. If this bid's utility value is equal to or lower than the weighted utility value of the received bid, we accept.

This is also done in order to help come to an agreement as time goes on. This does not mean we accept just any bid - our formula dictates that this strategy goes through when a factor beta, added to the received bid's value that is multiplied by a factor alpha, is greater than the next bid. Beta is 0.2, and alpha is 0.8 for best results - these numbers come from experimental outcomes using different inputs. Figure 5 illustrates the implementation of condition number two.

```
# Prediction based acceptance strategy
alpha = 0.8
beta = 0.2

next_offer = self.find_bid()
next_offer_value = float(self.profile.getUtility(next_offer))
strategy2 = (progress > 0.7
             and (((alpha * received_bid_utility) + beta) >= next_offer_value)
             and (received_bid_utility >= reservation))
```

Figure 5: Code for condition 2

Condition number three takes into consideration the parameters we wish to maximize: both agents' utilities and, thus, the Nash product and social welfare. Quite simply, we accept a bid if our Nash product at that stage would be above 0.4 or our social welfare would be above 1. Based on the experimental results of all groups and relevant literature on the Pareto Optimal Frontier, such as the referenced paper by Reyhan Aydogan, Tim Baarslag and Catholijn M. Jonker (2021), these values are self-evidently quite high, thus indicating an above-average solution to the negotiation issue.

Since we require two conditions to hold at the same time to accept a bid, this condition is designed to work later on in the negotiation - together with one of the time-based conditions - and has lower expectations in terms of the utilities and Nash product than condition number four, as can be seen in figure 6.

```
# Nash Product & Social Welfare based acceptance strategy
strategy3 = False
opp_value = float(self.profile.getUtility(self.last_received_bid))
our_value = float(self.opponent_model.get_predicted_utility(self.current_offer))
nash_product = opp_value * our_value
social_welfare = opp_value + our_value
if nash_product > 0.4 or social_welfare > 1:
    strategy3 = True
```

Figure 6: Code for condition 3

As can be seen in figure 7, condition number four dictates the results we'd like to obtain in the first 7 seconds of the round: Nash product above 0.4 or social welfare above 1.0 AND opponent utility value above 0.7. We do not wish to be too selfish this early on in the negotiation, rather, we're looking for best result both ways in this interval. Same as with condition number three, our desired values are self-evidently high and indicative of a great negotiation strategy.

If this does not come to pass, we know we should reach an agreement eventually - still above our reservation value - even if one side is dominant, then condition number three should hold: still good results, but perhaps more one-sided. As even more time passes, we become more and more lenient on the utility value we expect. In the last 0.03 of the round, as explained at the beginning, we take anything equal to or greater than our reservation value.

```
strategy4 = False
if progress < 0.7:
    opp_value = float(self.profile.getUtility(self.last_received_bid))
    our_value = float(self.opponent_model.get_predicted_utility(self.current_offer))
    nash_product = opp_value * our_value
    social_welfare = opp_value + our_value
    if (nash_product > 0.4 or social_welfare > 1) and opp_value > 0.7:
        strategy4 = True
```

Figure 7: Code for condition 4

5 Evaluation

In order to test the effectiveness and success of the negotiations made by AgentQT, we decided to test it in two different ways. The first would be against a magnitude of agents provided to us in the ANL2022 and CSE2310 folders in a tournament setting following SAOP. For this, four separate tournaments were run, each containing unique agents from either of the two folders alongside AgentQT. This would ensure a testing ground that could pit AgentQT against as many different negotiating strategies and techniques as possible. A more in-depth analysis of the success and workings of our agent was then conducted against five individual negotiators.

In each tournament/run, for each agent, the average utility, Nash product, and social welfare were recorded in order to allow a basis for comparison between AgentQT

and the others. The average utility was determined to be a good indicator of the average success of the bid. Since our acceptance and bidding models aim not to extort the opponent and instead reach a favourable agreement for both parties involved, the average Nash product and social welfare proved to be effective gauges to test the success of such a negotiator. Since AgentQT attempts to maximise multiple important values, future builds can strive to increase the effectiveness of the algorithms and allow for negotiations through fewer total bids made.

5.1 Tournament Analysis

The first method for evaluation is the tournament analysis. That is, running rounds of negotiation between AgentQT and a set of different agents. We run three tournaments and average values of the parameters that we monitor, specifically, both utilities, Nash product, social welfare, number of offers and agreement rates. The findings are illustrated in the following visual representations.

	utility	nash_product	social_welfare	offers
Agent11	0.742283	0.449263844	1.309535449	1736
Agent2	0.699667	0.390274853	1.283618235	1577.188
AgentQT	0.696421	0.407784725	1.267381154	1589.406
Agent14	0.671545	0.468109073	1.432123294	1049.313
Agent19	0.666617	0.264322424	0.939110547	97.25
Agent18	0.665059	0.43904259	1.343702491	931.125
Agent3	0.659222	0.443022697	1.337401781	2398.031
Agent22	0.499265	0.39651921	1.292896037	350.6563
Agent7	0.26526	0.236324454	0.924906814	642.0313

Figure 8: Tournament 1

	utility	nash_product	social_welfare	offers
MiCROAgent	0.533578	0.332633973	0.912192573	2076.536
RGAgent	0.518994	0.408494881	1.120630822	419.8571
AgentQT	0.493673	0.401717333	1.135368299	895.7857
LuckyAgent2022	0.472283	0.310148409	0.851923279	1009.179
SuperAgent	0.470608	0.341269667	0.945918979	1949.786
LearningAgent	0.450678	0.314088954	0.853076005	263.5357
GEAAgent	0.408112	0.326307541	0.932085583	22.57143
SmartAgent	0.135277	0.075533274	0.215211219	413.25

Figure 9: Tournament 2

	utility	nash_product	social_welfare	offers
DreamTeam109Agent	0.682235	0.400907213	1.177014548	2435.75
RGAgent	0.66166	0.420107413	1.189700949	436.35
Agent25	0.628426	0.300327456	0.938898416	2041.9
Agent3	0.626622	0.460111306	1.423983302	2866.65
AgentQT	0.581611	0.404003991	1.218807536	2861.5
AgentFish	0.490703	0.428873643	1.394108317	782.35

Figure 10: Tournament 3

	utility	nash_product	social_welfare	offers
DreamTeam109Agent	0.805469	0.465273965	1.38897576	1279.071
CompromisingAgent	0.688365	0.45735985	1.378885864	269.1429
BIU_agent	0.612767	0.341751735	0.993918483	518.9286
ChargingBoul	0.588587	0.360423103	1.029261979	1332.536
Agent007	0.547721	0.405398535	1.316767096	943.3571
Agent4410	0.544351	0.370073484	1.044319537	1054.786
AgentQT	0.536013	0.346733111	1.071675699	948.8571
AgentFish	0.469927	0.402519949	1.362595573	744.25

Figure 11: Tournament 4

If we average these values across all four tournaments for AgentQT and count the total number of successful negotiations, we obtain the following:

- Average Utility: 0.5769295
- Average Nash Product: 0.39005979
- Average Social Welfare: 1.173308172
- Average number of Offers made: 1573.8872
- Total number of Negotiations: 108
- Total number of Successes: 90 (83% success)
- Total number of Fails: 18 (17% fail)

In contrast, these are the average values for all the other agents that attempted to compete:

- Average Utility: 0.5507
- Average Nash Product: 0.3635
- Average Social Welfare: 1.0549
- Average number of Offers made: 952.9
- Average Percentage of Successes: 77%
- Average Percentage of Fails: 23%

To objectively analyse these results and further justify our choices, we must reference the document on The 13th International Automated Negotiating Agent Competition Challenges and Results: Aydoğan et al. (2023). Experimental results have been mentioned throughout our work as a basis for proving our agent's success.

As can be seen, all of our agent's parameters are comparable to some of the top competitors. It could be said that our agent is a jack of all trades yet fails to reach the heights of certain individual agents in certain parameters. Its consistency and completeness, if not supremacy, in getting high values for all the parameters considered is perhaps self-evident. Furthermore, AgentQT has a high agreement ratio and a number of offers that suggest a fruitful negotiation. Based on the opponents' average values in negotiating with our agent, too, it is clear that we handle negotiation in a more selfless manner than average - these relevant parameters being as respectable for us as we've come to expect from AgentQT.

5.2 Individual analysis

This section presents the outcomes of conducting individual negotiations with AgentQT against five distinct agents. The findings are illustrated through a detailed table mapping the results and accompanied by 3 negotiation graphs for visual representation of the most noteworthy findings.

	QT_utility	oponent_utility	nash_product	social_welfare	offers
DreamTeam109Agent	0.45616	0.95814	0.43706721	1.4143	4912
RandomAgent	0.760375	0.231384	0.175939312	0.991759	1078
ThirdAgent	0.825825	0.31993	0.26420686	1.1457559	170
RGAgent	0.502975	0.880748	0.44299512	1.38372	534
ProcrastinAgent	0.4561621	0.95814	0.437067218	1.414302	5312

Figure 12: Individual Runs

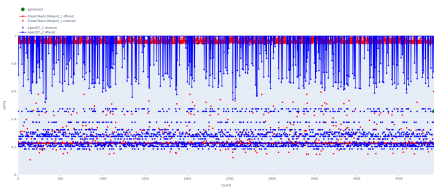

Figure 13: AgentQT vs. DreamteamAgent109

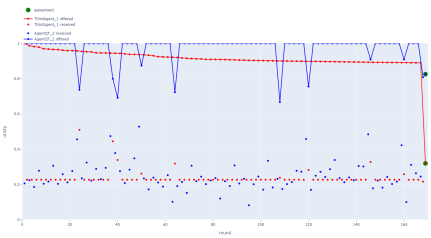

Figure 14: AgentQT vs. ThirdAgent

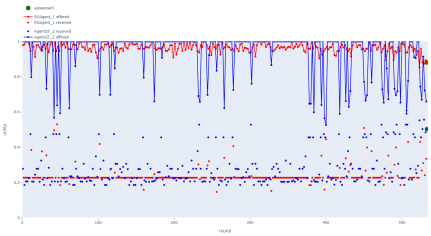

Figure 15: AgentQT vs. RGAgent

In conclusion, the individual analysis demonstrates AgentQT's versatility and effectiveness when engaged with diverse agents. Notably, AgentQT exhibits a strategic flexibility, emphasizing either its own utility or prioritizing the overall Nash product and social welfare, thereby showcasing its adaptability and negotiation prowess through the Trade Off Strategy.

5.3 Deficiencies and Future Improvements

During the individual rounds, a notable challenge emerged as AgentQT encountered difficulties when negotiating with ProcastinAgent, resulting in a failure to reach an agreement in 3 out of 5 instances. The same can be observed when running AgentQT against Agent25 from CSE3210. The cause is estimated to be the agents not being flexible at all up until the very end of the negotiation, which causes our agent to reject the offers due to them being below the reservation value. A possible fix could be the decrease in reservation value. However, this would yield negative results for our agent when run against certain exploitative agents. For future builds, experiments can be carried out to determine the optimal reservation value for our agent.

Additionally, it was observed that AgentQT exhibits a high average number of offers, with an average of 1573.8872 offers made during the tournaments. Future versions could definitely see an implementation of the bidding and acceptance models to reduce this large number of offers. As seen from the tournaments, the average number of offers made from the opponents was about 950, indicating that major improvements can be made here.

As mentioned above, the main drawback with the current implementation for the agent is although it attempts to be a jack of all trades and maximise multiple values, it ends up not performing as high as some other agents across individual values such as utility gained. In the future, this can definitely be improved upon, such as developing a model that allows the agent to obtain higher utility while still seeking to maximise the Nash product and social welfare statistics.

Furthermore, the current implementation of AgentQT also adapts a somewhat simplistic opponent modelling system. Future versions of the agent could indeed have a much more advanced system, allowing the agent to be harsher about the concessions it makes. This would, in turn, lead to higher utility gained from the offers.

Lastly, while implementing a multi-party negotiating agent was beyond the scope of the assignment, it is definitely a feature that can be added to AgentQT in the future. This could allow for even more chances of observations and evaluations in a tournament setting with other mulit-party negotiating agents.

6 Conclusion

All in all, our experimental results show that our agent consistently handles negotiation with other agents in an effective manner. We obtain values in proximity to the Pareto Optimal Frontier and compare them to the top groups in several of the key parameters, such as individual utilities, Nash product and social welfare; and our rate of agreement does not have to suffer as a result. The theory of negotiating agents suggests that it is impossible to make any bid that would improve both agents' value beyond the Optimal Frontier.

On this frontier, multiple points have been identified as appropriate solutions, such as the maximal Nash product solution and the egalitarian solution. One stance is presented in the paper of van Damme (1986) in the Journal of Economic Theory; however, for the purpose of innovation in our project, we adapt the Kalai-Smordinsky approach and use the average of the agents' utilities in a weighted formula taking into account the best bid.

In summary, our research, AgentQT and it's evaluations underscores the potential for negotiating agents to navigate complex scenarios effectively, paving the way for enhanced cooperation and resource allocation in diverse contexts.

Individual Contribution

Each member of the team contributed equally to all aspects of the project, from the initial analysis and discussion of the project objectives to the discussion and planning regarding the negotiation mechanism, the implementation of the negotiation mechanism itself, the discussion and analysis regarding the findings, and lastly, finalizing the report.

All activities were carried out in the presence of all teammates on or around university premises.

References

Aydoğan, R., Baarslag, T., Fujita, K., Hoos, H. H., Jonker, C. M., Mohammad, Y., & Renting, B. M. (2023). The 13th international automated negotiating agent competition challenges and results. In R. Hadfi, R. Aydoğan, T. Ito, & R. Arisaka (Eds.), *Recent advances in agent-based negotiation: Applications and competition challenges* (pp. 87–101). Springer Nature Singapore.

Catholijn M. Jonker, T. B., Reyhan Aydogan. (2021). *Negotiating agents: Reader*.

Koen Hindriks, D. T., Catholijn M. Jonker. (2011). Let's dans! an analytic framework of negotiation dynamics and strategies. *Web Intelligence and Agent Systems: An International Journal*, (9), 319–335. https://doi.org/10.3233/WIA-2011-0221

P. Faratin, N. J., C. Sierra. (2002). Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, *142*(2), 205–237. https://doi.org/10.1016/S0004-3702(02)00290-4

Raiffa, H. (1982). *The art and science of negotiation*. Harvard University Press.

van Damme, E. (1986). The nash bargaining solution is optimal. *Journal of Economic Theory*, *38*(1), 78–100. https://doi.org/10.1016/0022-0531(86)90089-X