CS 551 Project Group 4 Agent Report

Jun 10, 2022

Arash Mehrabi
Mert Erkol

Introduction

In today's world we negotiate every day without being noticed. Every action or dilemma in our life is a negotiation and sometimes these dilemmas or actions can be critical and we have to choose the best from some list of possibilities or we have to decide something on a project that we are working on as a team and we as a team have different opinions and preferences and what happens if we can't decide on some single solution. The examples that are mentioned are real world problems today that can be solved with negotiation agents with the help of artificial intelligence techniques and possibilities.

In this project, we designed and implemented an agent that uses frequency modeling for its opponent modeling section, and a time and behavior dependent bidding generation strategy. We also used the AC Next as the acceptance strategy. We concluded that our proposed bidding strategy works better, especially against boulware like agents. Our agent is a fair agent who doesn't selfishly only takes its own utility into account. It also pays a lot of attention to opponents utility. However, you can't bully this agent because it also uses a tit for tat algorithm and tries to be nice with nice opponents and rough with unfair opponents.

High Level Design

Opponent Modeling:

For opponent modeling, we used the frequency opponent modeling described in [1]. The idea behind this technique is like this: if the value of an issue changes often, then the issue gets a lower weight. Furthermore, if a value occurs many times in the agent's offers, we conclude that the value is more preferred.

The *update* function from the *OpponentModel* class is called whenever a bid is received from the opponent. It checks the received bid with the previous bid that was received and increases the weights of issues whose value have not change between the two bids.

Each object of the *Issue* class has an attribute called *num_occurences* which is a dictionary that counts the number of occurrences of each value in opponents offer. From the *update* function we call the *update* function of this class to increase the number of occurrence of a value that occurred in opponent's new bid.

After we updated the *num_occurences* for each value of the opponent's new bid, we calculate the weight of each value in the domain of each issue by dividing the number of of occurrences of the value to the greatest number of occurrences in the issue.

Bid Generation Strategy

The algorithm we used for the bid generation strategy was a customized time dependent algorithm. At each time, the algorithm selects a bid that is greater than than a lower bound utility based on the opponent's preferences. So, for example, if the utility be 0.7, the algorithm first finds all the bids which have a utility greater than 0.7. Then, from those bids, it selects the bid that is mostly preferred by the opponent, based on the estimation of the opponent model that we developed.

The lower bound that we mentioned in the previous paragraph, decreases based on time. (Please see Figure 1). For the interval t: 0 <= t < 0.3, it starts from 0.9 and decrease linearly until 0.7. Then, inside the interval 0.3 <= t < 0.6, the utility is a constant value 0.7. After that interval, the utility again decreases linearly from 0.7 to 0.4 in the remaining interval (t: 0.6 <= t < 1).

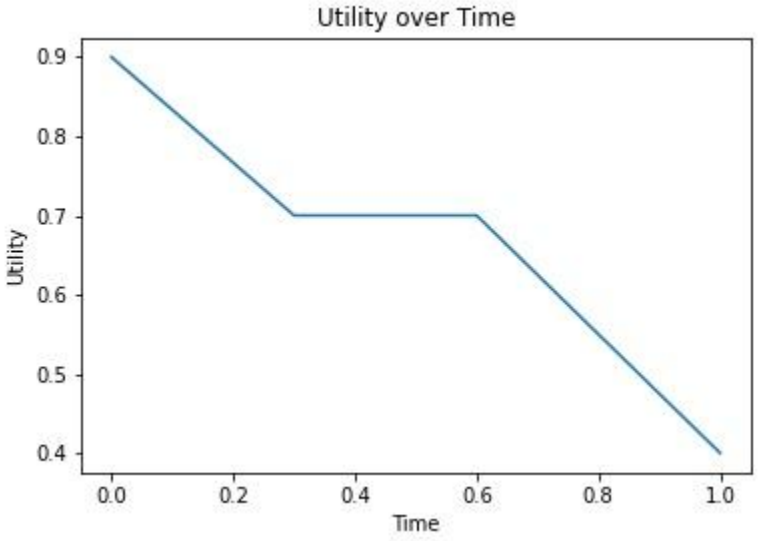

Figure 1: Behavior of utility variable over time

We also find out that in some cases the algorithm sends the same offer several times because it thinks that it is the best offer for the opponent. For that reason, we set a limit on the number of

times an offer can be send. This is a hyperparameter and we decided to set it to 5; therefore, no offer can be send more than 5 times when negotiating with an agent.

The *get_bid_greater_than* function from the *utils* module is responsible for selecting the bid that is greater than the given utility. From the selected bids, it chooses the bid that is most preferred by the opponent and it prevents sending a bid more than 5 times.

Acceptance Strategy

Acceptance strategy is responsible for deciding whether the received bid should be accepted. For this, we used the AC next condition, which accepts the received offer if its utility be greater than the next offer of our agent.

Performance Evaluation

Since our agent was time dependent, it was defeated by boulware like agents (Figure 2). Since no one likes getting bullied (!), we changed our model to defeat the boulware like agents while keeping in mind to not become a one.

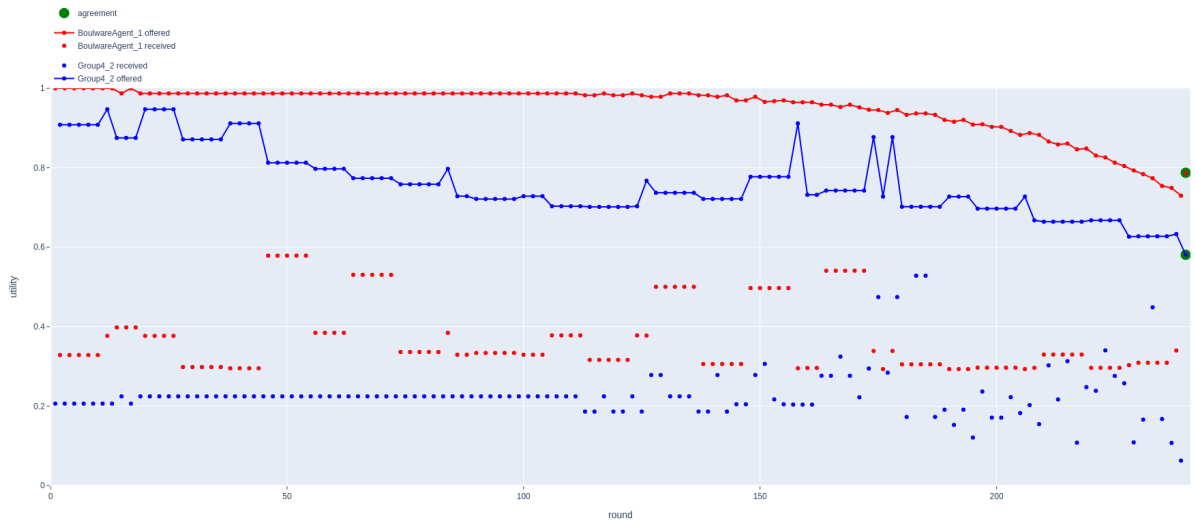

Figure 2: Our agent vs boulware agent

For this reason, we tried to change our bidding strategy so that it considers the behavior of the opponent as well. The idea is to average the utility of the last 3 offers of the opponent and generate an offer according to that. So, if the average be 0.3, our behavior dependent bid generation algorithm will select a bid greater than 1-0.3=0.7 to offer. It is like tit for tat, if the opponent's generated offers' mean be 0.6, our behavior dependent bid generation algorithm will

select a bid greater than 1-0.6=0.4. As much as opponent concedes, we will act generate more generously.

Then, we will have two bids to offer. One, that is resulted from the time dependent algorithm, and the other that is product of the behavior depent algorith. We will select the one that has the greater utility for us. This way, if our time-dependet algorithm produces a bid with the utility of 0.48; but, the mean of last 3 bids of the opponent be 0.3; therefore, the behavior-dependent algorithm return a bid of utility greater than 0.7, our bid generation strategy will select the behavior-dependent algorithm's bid with utility of 0.7.

You can see the result of a negotiation of our agent with the same boulware agent on the same domain in Figure 3. As it is shown in the figure, our agent defeated the boulware agent with this new bidding strategy.

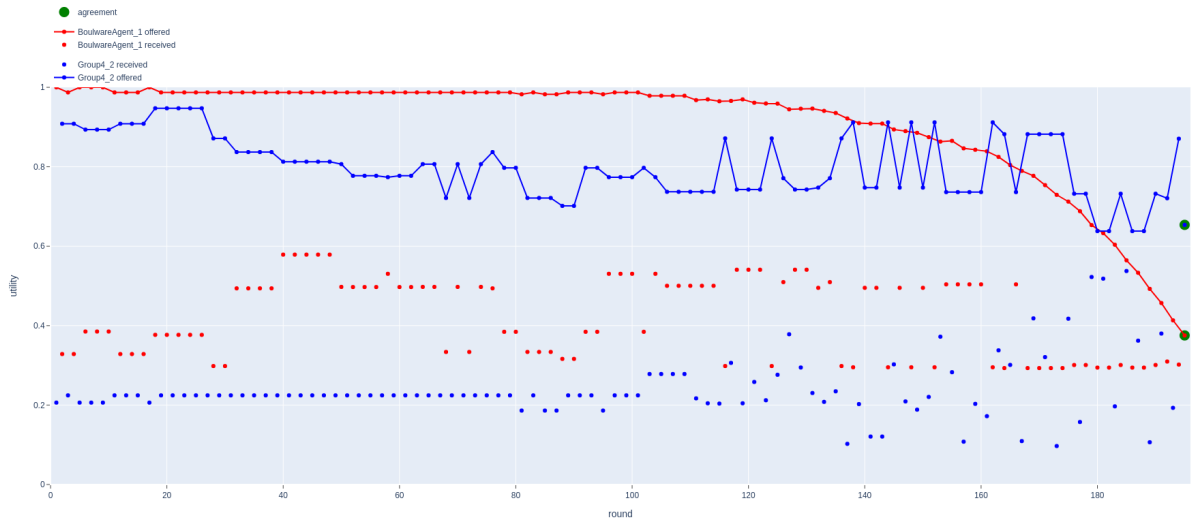

Figure 3: Our agent with time and behavior dependent bidding strategy vs boulware agent

Our model also shows an acceptable performance against the hybrid agent. (Figure 4).

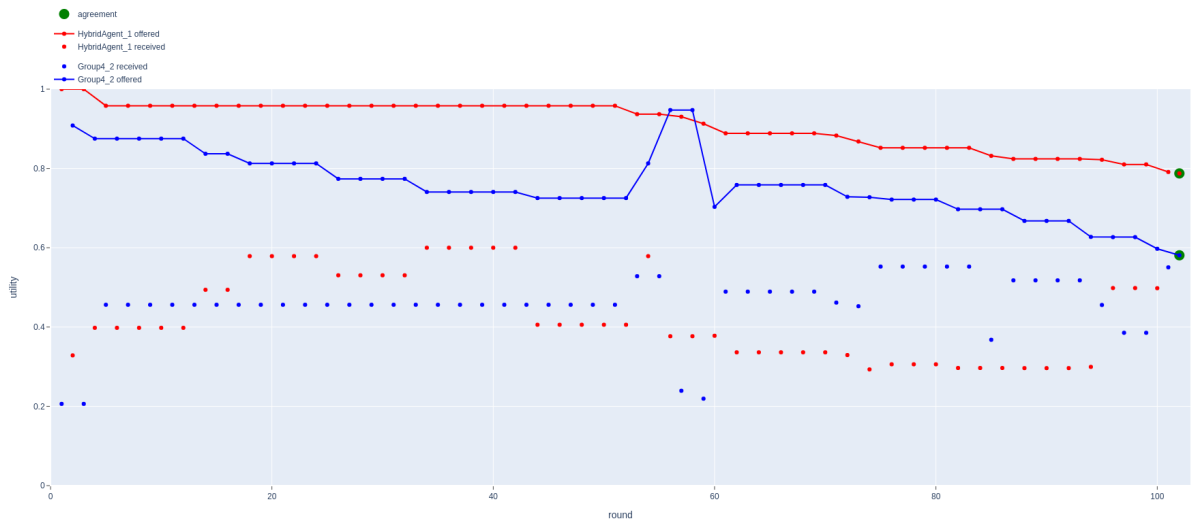

Figure4: Our agent with time and behavior dependent bidding strategy vs hybrid agent

And it can also defeats the time dependent agent (Figure 5).

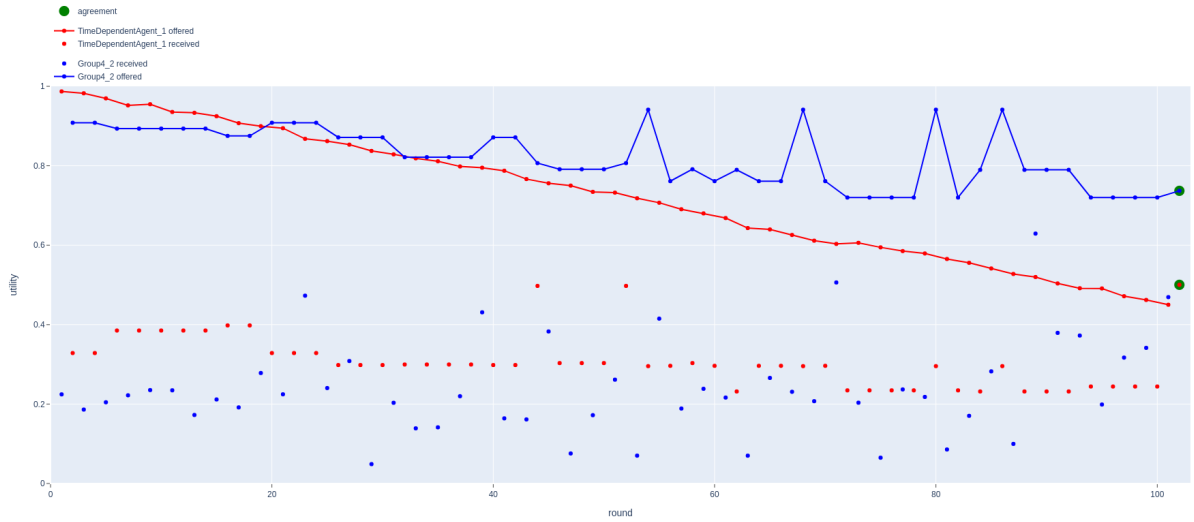

Figure5: Our agent with time and behavior dependent bidding strategy vs time dependent agent

Details on behavior dependent bidding strategy

Now it is a good place to explain the details of behavior dependent bidding strategy. If time be lesser than 0.7, we use this strategy with the chance of 0.5. The idea behind this is to forgive the opponents bad behavior with the chance of 50% before the time 0.7. After time passes 0.7, we use this strategy ⅔ of time; however, we don't use time-dependent strategy the rest of times. At the rest of times, we generate a bid with utility greater than 0.7.

Comparison

The following is the result of running a tournament on domain01 and domain05 with the mentioned agents (Table 1).

	avg_utility	avg_nash_product	avg_social_welfare	avg_num_offers	count	agreement	failed	ERROR
BoulwareAgent	0.762	0.467	1.426	4047.0833	24	24	0	0
LinearAgent	0.660	0.487	1.449	1930.333	24	24	0	0
HardlinerAgent	0.620	0.292	0.914	4674.833	24	15	9	0
ConcederAgent	0.602	0.482	1.443	837.666	24	24	0	0
Group4	0.579	0.412	1.149	158.541	24	19	5	0
HybridAgent	0.575	0.325	0.960	283.25	24	16	8	0
RandomAgent	0.484	0.427	1.226	3272.708	24	20	4	0

Table 1: The result of a tournament

Conclusion

In this project, we designed and implemented an agent that uses frequency modeling for its opponent modeling section, and a time and behavior dependent bidding generation strategy. We also used the AC Next as the acceptance strategy. We concluded that our proposed bidding strategy works better, especially against boulware like agents. Our agent is a fair agent who doesn't selfishly only takes its own utility into account. It also pays a lot of attention to opponents utility. However, you can't bully this agent because it also uses a tit for tat algorithm and tries to be nice with nice opponents and rough with unfair opponents.

During the implementation we learned to code as a team and used several different negotiation techniques. Also, we had the chance to do some literature review and saw some existing agents. In the future a much more advanced and simple graphical user interface would be a perfect solution to support human negotiations.

References

1. Tunalı, O., Aydoğan, R., Sanchez-Anguix, V. (2017). Rethinking Frequency Opponent Modeling in Automated Negotiation. In: An, B., Bazzan, A., Leite, J., Villata, S., van der Torre, L. (eds) PRIMA 2017: Principles and Practice of Multi-Agent Systems. PRIMA 2017. Lecture Notes in Computer Science(), vol 10621. Springer, Cham. https://doi.org/10.1007/978-3-319-69131-2_16