Agent5

CS462 Project – Group 5's Agent

Batuhan Erden, Ekrem Cetinkaya, and Daniyal Anwar

Abstract

In this analysis report, we will describe *Agent5*'s strategies and *Agent5*'s performance against *ParsAgent*, *Atlas3* and *RandomDance*. We will also describe how did improve *Agent5*'s performance (Part 1) by improving *Agent5*'s strategies.

Introduction

Automated agents can be used a long side the human negotiator undertaking on a negotiation task. They can temper some of the endeavors necessary of people during negotiations and also help people that are less qualified in the negotiation process. There may come a day in near future where automated negotiators may replace the human negotiators. Another possibility is for people can also use these agents as training tools before they actually perform a task. Hence, level of success in creating an automated agent with the capabilities of negotiating will have great advantages and implications.

There are many different successful strategies that already exist in the science. To compare the results of such agents vary from environment to environment like Genius in this case. One can use many different theories, strategies and models for making an agent.

With Genius in place, we organized *Agent5* with the aim of coordinating the issue and automating the result for multi-lateral negotiating issues with its multi-lateral negotiating strategies. Our agent we believe is qualified and good addition to the existing negotiating competitions.

Headers

- About Agent5
- Concession Strategy (Boulware Level)
- Risk Function and Generating Fake Bids
- Updating Threshold Value over Time
- Opponent Modeling
- Generating a Bid
- Accepting an Offer
- Performance
- Conclusion

About Agent5

In this report we will discuss about *Agent5* in detail. *Agent5* like many other agents is a multi-lateral negotiator agent that will follow "Stacked Alternating Offers Protocol for Multi-Lateral Negotiation. In this competition, we only consider multi-lateral negotiations, which is a negotiation in more than two parties. The parties go and negotiate over different issues, and every issue has an associated range of alternatives or values. A negotiation outcome maps every issue to a specific value, and the set Ω of all possible outcomes is called the negotiation domain. The domain is common knowledge to the negotiating parties and stays fixed during a single negotiation session.

Agent5 is an agent that focuses on different areas. *Agent5* not only focuses on its own utility but also focuses on the utilities of the opponents as well. The utility maximization is not the only thing our agent focuses on. It also focuses on the social-welfare as that is one of the key aspects in any negotiation. Last but not the least, *Agent5* cynosures about the agreement rate of the opponent and itself again as that is the point where a negotiation round is finishing hence is quite important.

Agent5 has many different negotiating strategies that are mainly centered towards the Opponent Modeling that is our main priority in any negation. *Agent5* uses different strategies for different opponents depending on their deportment. Our agent tries its best to bring the best possible solution or optimal outcome for all the negotiating parties with maximized benefits for all the parties and tries to be fair in every round or the negotiation.

Agent5 strategies and performances are tested against many different agents in the test and trial. The detail strategies and performances against *ParsAgent*, *Atlas3* and *RandomDance* are delineated.

Concession Strategy (Boulware Level)

In Agent5's Opponent Model class, there is a concept called *Boulware Level*. *Boulware Level* is the level of Boulwareness (How much boulware the opponent is) of the opponent. *Boulware Level* is used for modifying the behavior of our agent. If the opponent doesn't concede, we shall not concede too.

"The higher the Boulware Level is, the more Agent5 increases its threshold."

- When it's 0, Agent5's threshold value remains unchanged.
- When it's 5 (MAX), Agent5's threshold value is 1.

The *Boulware Level* is decided by checking the offers of the opponent. If the utilities of the offers (according to our utility function) that are offered by the opponent are low, then the *Boulware Level* will be high accordingly.

Calculating Boulware Level (*Formula Box – a*)

```
Boulware Multiplier = 10
Edge of Conceding = threshold * 0.8
Final Received Utility = Average Utility(Last 2 Received Bids)

if (Utility(Final Received Bid) > Edge Of Conceding)
        Boulware Level = 0
else
        Boulware Level = (int) (Edge Of Conceding - Final Received Utility) * Boulware Multiplier)
```

Boulware Level changes as the opponent offers. **C** is the constant value that makes sure that threshold value is **1** if the Boulware Level is at maximum (**5**). No matter how high is the threshold value, the new threshold value will not be higher than 1 with the help of the **C** constant.

Calculating New Threshold (*Formula Box – b*)

```
Maximum Boulware Level = 5
C = (Maximum Boulware Level * threshold) / (1 - threshold)

New Threshold = threshold * (1 + Boulware Level / C)
```

This *Boulware Level* will then be used to update the threshold value while generating a bid.

Risk Function and Generating Fake Bids

Agent5 generates fake bids to make it hard for the opponent to predict our preferences. The frequency of generating fake bids is decided by the *Risk Function*. It will always be risky since the opponent can accept our fake bids. We thought that our agent should take some risks in order to be successful.

Risk Function, $\mathbf{F = C / 2^P}$, where F is the Round Numbers to Fake (Agent will fake in every F rounds), C is the Risk Constant, in our strategy, we set C = 100,000, and P is the Risk Parameter that takes a value between {1, 2, … , 10}, in our strategy, we set P = 5 because we want our agent to be both aggressive and defensive (1 is the most defensive parameter and 10 is the most aggressive parameter).

Updating Threshold Value over Time

We store our threshold value as a global variable and it's 0.8. At the beginning of each negotiation, we multiple our threshold value with 1.125 so that it becomes 0.9. There are two concepts that affect our threshold value:

- **Time to get Almost Mad:** Our agent becomes almost mad at the second half of the negotiation. When our agent is almost mad, we just multiply our threshold value (0.9) with 0.95 so that it becomes 0.855 and this value is our current threshold.
- **Time to get Mad:** Our agent finally becomes mad in the last 20% of the negotiation. When our agent is mad, we just multiply our threshold value (0.9) with 0.9 so that it becomes 0.81 and this value is our current threshold.

The main reason why we decrease our threshold over time is because we seek for an agreement because if we can't agree with the opponent with a high threshold, it means that the threshold value should be lower so that we can reach an agreement.

Opponent Modeling

In Opponent Model class, we simply model our opponent's preferences. There is a preferences list in Opponent Model class. For each item in the domain, the number of occurrences of that item is stored in preferences list. We extract how many times the opponent prefers an item from the opponent's bid and add it to the preferences list. In our preferences list, we are storing Preference objects that consist of *Issue, Value and Count* as mentioned below.

```
class Preference {
        Issue issue;
        Value value;
        int count;
}
```

For example, imagine that the opponent offers Beer 5 times. At that instance, there will be only one Preference object for Beer in the preference list that stores *Issue (Drinks), Value (Beer), and its Count (5)*.

- **Computing the Most Preferred Bid by the Opponent**

We also compute the *Most Preferred Bid* by that opponent by using the preferences list. We construct a bid and for each issue, we add the value with the highest count. For example, in the table below, the most preferred items can be seen.

Issue **Value** **Count**	Drinks Beer Only 1089	Music DJ 1061	Food Chips and Nuts 1056	Location Party Room 985	Cleanup Specialized Materials 920	Invitations Custom, Handmade 651
Issue **Value** **Count**	Drinks Handmade Cocktails 18	Music MP3 42	Food Finger-Food 28	Location Ballroom 100	Cleanup Water and Soap 131	Invitations Photo 226
Issue **Value** **Count**		Music Band 4	Food Handmade Food 21	Location Party Tent 22	Cleanup Hired Help 32	Invitations Custom, Printed 151
Issue **Value** **Count**			Food Catering 2		Cleanup Special Equiment 24	Invitations Plain 79

So, in this case, our *Most Preferred Bid* is:

<Beer Only | DJ | Chips and Nuts | Party Room | Specialized Materials | Custom, Handmade>

- **Predicting the Weights of Each Issue in Opponent's Domain**

We also predict the weights of each issue in opponent's domain in order to manipulate the most preferred bid by the opponent in a way that we get a high utility from that bid and opponent likes that bid.

Issue	Drinks	Music	Food	Location	Cleanup	Invitations
Value	Beer Only	DJ	Chips and Nuts	Party Room	Specialized Materials	Custom, Handmade
Count	1089	1061	1056	985	920	651

Now that we get the most preferred values by the opponent for each issue, we can now calculate the weights of each issue. To do that, we just normalize the counts. In this case, the weights are:

Total Sum = 1089 + 1061 + 1056 + 985 + 920 + 651 = 5762
Weight of Drinks = 1089 / 5762 = 0.1890
Weight of Music = 1061 / 5762 = 0.1841
Weight of Food = 1056 / 5762 = 0.1833
Weight of Location = 985 / 5762 = 0.1709
Weight of Cleanup = 920 / 5762 = 0.1597
Weight of Invitations = 651 / 5762 = 0.1130

After we calculate the weights for each issue, we pick the issues with the 1st and the 2nd lowest weights. If the difference between the 2nd lowest weight and the 3rd lowest weight is smaller than or equal to 0.02, we also pick the issue with the 3rd lowest weight, and it goes on like that until the difference is greater than 0.02.

Then for each picked issue, we change the values of the *Most Preferred Bid* by the opponent and add to the list called Acceptable Bids. For example, image that one of our picked issues is cleanup. We construct our bids by only changing the values of cleanup and add it to Acceptable Bids. Then, we are going to sort and use Acceptable Bids while generating a bid.

Generating a Bid

Firstly, there are 3 Opponent Model instances. One is for the preferences of the 1st opponent, one is for the preferences of the 2nd opponent, and the other one is for the preferences of both of the opponents (Mixed Preferences).

- **In the first 5% of the negotiation**

Our agent generates its *Second Best Bid* in the first 5% of the negotiation in order to stay still and watch how opponent acts. *Second Best Bid* is calculated using SortedOutcomeSpace. The reasons why our agent doesn't generate its *Best Bid* is because we don't want to give the information about what we like the most to the opponent.

- **Generating Fake Bids**

By using the formula we described in **Risk Function and Generating Fake Bids**, The frequency of generating fake bids, F = 3125. This means that our agent will generate fake bids in every 3125 rounds and it will fake 10 times. For example, it will fake in round 3125, 3126, 3127, 3128, 3129, 3130, 3131, 3132, 3133, 3134, 3135.

Our agent will not generate fake bids in the last 10% of the negotiation.

Calculating when to generate fake bid (*Formula Box – c*)

```
if (Number of Rounds Passed % F <= 10 && Current Status <= Negotiation Limit * 0.9)
        Generate Fake Bid
```

Our agent then generate fake bid by generating a random bid with a utility higher than the 80% of the threshold.

- **Generating a Bid using Opponent Modeling**

At first, we choose a random value among {0, 1, 2} to decide which opponent model instance we are going to use to generate a bid according to that opponent's preferences.

  - If the **random value is 0**, we generate our bid using the knowledge we get from the **preferences of the 1st opponent**.
  - If the **random value is 1**, we generate our bid using the knowledge we get from the **preferences of the 2nd opponent**.
  - If the **random value is 2**, we generate our bid using the knowledge we get from the **preferences of both opponents**.

At second, we first change our threshold value to the new threshold value obtained using Boulware Level **(Formula Box – b)**.

Then, we use **Time to get Almost Mad** and **Time to get Mad** concepts that are described above to decide our current threshold value.

❖ **If 99% of the negotiation is reached**

If the utility of the best-received bid is higher than or equal to the current threshold, **our agent generates the best-received bid**.

❖ **Else if the agent is almost mad**

First, we are taking the acceptable bids that are calculated using the weights of the opponent. Then, we are sorting them in a way that the bid with the highest utility (according to our utility function) is located at the first index. Then, we get the acceptable bid at *shiftBids* (is a unique number, initially is 0, for all opponent model instances) index and we increment *shiftBids*.

- If the utility of the chosen acceptable bid is higher than or equal to the current threshold, **our agent generates the chosen acceptable bid**.
- Else, *shiftBids* becomes 0 because any bid that is located at the right of the *shiftBids* is also less than the current threshold. What this means is that our agent don't have to consider that bids.

❖ **Else**

Our agent generates a random bid with a utility higher than the current threshold.

Accepting an Offer

Agent5 simply **accepts an offer** if the utility of that offer (according to our utility function) is higher than the current threshold value. *Agent5* won't accept anything lower than the very first threshold value.

Performance

- **Agent5 Part1 vs. Agent5 Part2**

In Part1, we sometimes couldn't reach an agreement with agents. So we changed now strategy a bit. Now, our agreement rate is always 98-99%.

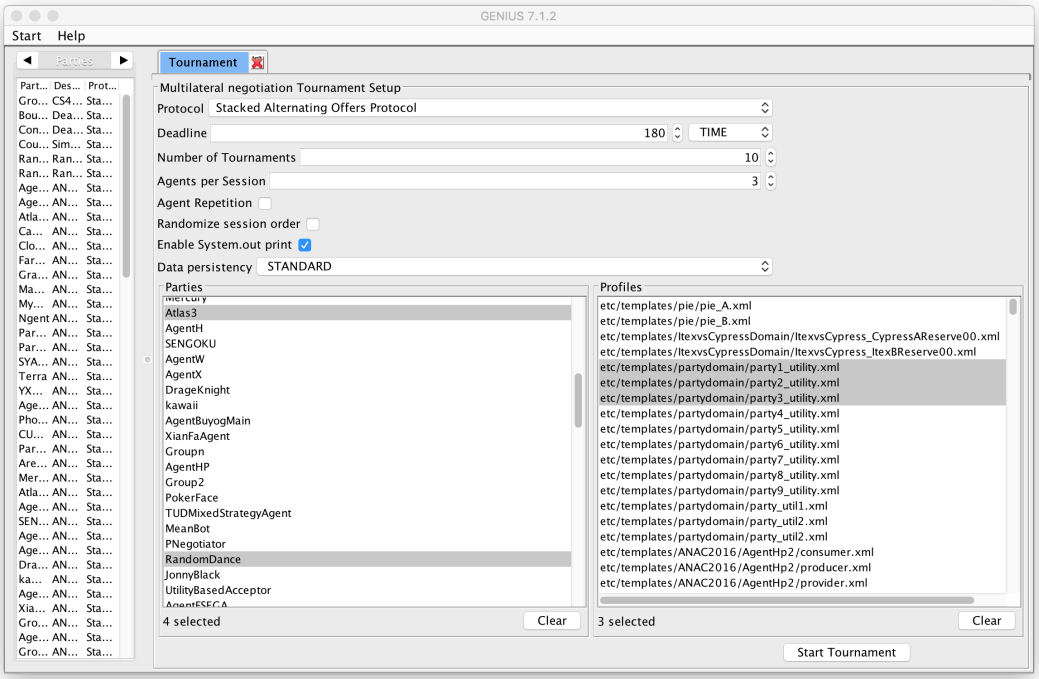
In Part1, we were increasing the threshold value over time. What it means is that when our agent becomes mad or almost mad, we were increasing the threshold value. In Part2, we have changed that. Now, we are decreasing the threshold value over time.

In Part1, if we couldn't reach an agreement until the last 5% of the negotiation, we were offering bids with a low utility to reach an agreement. In Part2, we eliminated that strategy. Instead, we have developed better strategies.

In Part1, we were only using our Opponent Model at the end of the negotiation. In Part2, we stopped doing that and started using it more often. This is the improvement that resulted a significant increase in our win rate.

In Part1, there was only one Opponent Model instance that stored the preferences of both of the opponents. We realized that we were doing something wrong because our opponent weren't accepting our offers. So in part 2, we created 3 Opponent Model instances. One is for 1st opponent, one is for 2nd opponent, and the other one is for 1st and 2nd opponent (both). In doing so, we are now able to generate bids that opponent likes.

- **Performance Tests of Agent5 Part2**



We ran the tournament as above in order to test our agent's performance. The results were encouraging. After improving our agent, we started winning against the agents that were defeating us before (Part 1). Here are the results:

Agreement Rate
98.02%

Average Social Welfare
2.2959

Group5	→1st Place
Average Utility	Standart Deviation
0.8242	0.1385

Atlas	→4th Place
Average Utility	Standart Deviation
0.6989	0.1381

Pars	→3rd Place
Average Utility	Standart Deviation
0.7519	0.1049

RandomDance	→2nd Place
Average Utility	Standart Deviation
0.7864	0.1489

Conclusion

This report shows us the automated negotiating agent competition. Based on the process, the submissions and the closing session of the competition our aim has been accomplished. The agent is ready to work and negotiate with different agents under different circumstances. The *Agent5* is an adaptive agent it adapts different strategies for different agents it sees if an agent is conceding or refraining from conceding. Its adaptive nature is one of the innovative implementations among the agents this makes it bit unique. The agent deals very fairly among the different agents for the opponents and itself as gives the maximum utility to everyone. It's kind of a like win-win situation for everyone. The social welfare is also taken care by the agent, which shows it is proficiency.

Overall our experience as a team was pretty good in developing this agent for a tournament. This agent taught us many things about artificial intelligence and some, what machine learning. Building an agent was a no walk in the park had to burn the midnight oil for it. Most important part in making this agent was making the or rather coming up with new and different strategies for new and different situations. This was something that involved opponent modeling a lot and a need to understand opponent in a much better way. The strategies we came up with are no different then what a human mind thinks or is capable of thinking just simple negotiation strategies with much complex calculations. Not only have we learnt from the strategy concepts we made for *Agent5*, we have also learned a great deal of understanding in the correct setup of a negotiation competition.

The *Agent5* has the capability of being used in real-life in different department maybe something like a **NEGOLATOR** (Negotiating Calculator) that could be used for future negotiations between different parties. This could be a small device where one can enter the bids or the weights or whatever he feels like to get the maximum utility maximum social welfare value or whatever he feels would help his case in making the decision faster. The negotiations with the help of **NEGOLATOR** would help the human negotiations and would increase the speed of negotiation.

This could possibly replace human negotiation with automated negotiation strategies in the near future.