Report AI

Group 9

Nathan Buskulic
4947916
Tu Delft

Tomas Thorbjarnarson
4917480
Tu Delft

Ruth Guimarey Docampo
4935462
Tu Delft

Thorunn Arna Omarsdottir
4917499
Tu Delft

Rukai Yin
4837371
Tu Delft

1 INTRODUCTION

The first goal of the assignment was to design and implement a negotiation agent to negotiate for us the best possible arrangement in a party-throwing negotiation session. Of course, the final goal was that our agent could negotiate on all possible negotiation domains and even under uncertainty. In order to achieve that, we had to use the BOA framework which is a simple way of representing an agent through individual components (Bidding strategy, Opponent model, Acceptance strategy). In order to know how well our agent performs, we have to test it against all other agents of the assignment and compare these results to the expected one. Our aim was to be able to map a considerably good preference profile for the opponent while accepting bids that are higher than our time-dependent acceptance threshold. Also trying to throw our opponent of a bit by not bidding our highest utility right away, but build up to it in the first third of the negotiation time. We will get to know more about this agents behaviour in the next sections of the paper.

2 STRUCTURE OF THE PARTY

As stated in the introduction, we constructed our agent using the BOA framework. Using this framework gave us an already existing high-level structure such that we only had to tweak each component but not the entire structure. The BOA framework uses three distinct components to represent an agent: the Acceptance strategy, the Bidding strategy and the Opponent model. The software that we are using (Genius [1]) is linking all of these components together in a simple way.

First, when our agent receives a bid, it calls the function *determineAcceptability* of the Acceptance strategy that says if we accept or not the offered bid with regard to our decision function (see section 3.1). Depending on the result of that function, two paths can be taken: either we accept the bid and the session ends here, either we refuse the bid and we need to offer a new bid.

The first thing to do after we refused a bid is to update our opponent model by using the function *updateMode*l of the OpponentModel component. For this agent, we decided that we should always update the model, thus the function *canUpdateModel* will always return *True* and the opponent model will always be updated accordingly to our function (see section 3.4)

Once we updated the opponent model, we need to offer a new bid. The software will call the function *determineNextBid* of the Bidding strategy which will provide a bid that we estimate good for us and the opponent. In order to know what could be good for the opponent, this function calls the function *getBid* of the Opponent Model Strategy which returns exactly what we want: a bid that is good for both agents. Our agent will also use this function when bidding the first time, thus the function *determineOpeningBid* will just call *determineNextBid*.

3 EXPLANATIONS

The following are descriptions of our thought processes and decisions when implementing the specific components of the BOA framework.

3.1 Acceptance Strategy and decision function

To decide how we wanted to have our acceptance strategy (AS), we first discussed what exactly an AS should accomplish and how it applied to our negotiation environment. Generally, an AS should decide if an opponent's offer results in a utility for our agent that is "good enough" to accept so the negotiation can stop. If it is not, our agent rejects and the negotiations continue. Consequently, if no bid is deemed acceptable, there are cases where no agreement is reached.

When negotiating, an agent may have a reservation value so it is fine that an agreement is not reached. In this case, it should not accept any bid that is less than its reservation value. However, since in our environment our agent has no reservation value, we decided that we should always strive to reach an agreement. The downside of this is that when bidding against an agent that only bids its optimal bid, our agent will always lose.

There was some almost philosophical discussion around this. Is it better to have an agreement where our agent has, let's say, utility $UA = 0,3$ and the opponent has $UO = 1,0$ or to reach no agreement so $UA = UO = 0,0$? We had no concrete answer and felt that in a tournament environment, having some utility is always better than having no utility. This would then also influence decisions made for our other strategies.

To do this we first decided that any bid by the opponent that results in utility higher than $UA = 0.8$ is accepted by default. Otherwise, we accept bids according to a function of time. We normalize the rounds or time left so that they are on a scale of 0 to 1, where 0 is no time left and 1 is all the time left. We started off by accepting bids that have a utility higher than the time left. We realized that this function falls off too quickly so we were accepting bids of too low utility. To make this a bit smarter we wanted the time function to be a root of the time so that it decreases slower. This can be visualised as: if $(U \geq min(0,8, T^x)$, where $0 \leq x \leq 0,5$ and T is

the normalized time left, then accept the bid. Deciding how big x should be, and therefore how fast our threshold descends took some tweaking and testing. In the end, we decided that $x = 0,5$, that is the square root of x, but we also offset our function by a value of 0,35. This means that for the first 80% of the time (rounds), our threshold is set at 0,8. In the final 20%, the threshold changes to $x^{0.5}$ and descends much quicker. This results in more bids being accepted in the final 20% of the negotiation session.

This strategy has some advantages and disadvantages. Our agent is quite stubborn, and for the most part, won't accept any bids that are less than 0,8. This can be viewed as both an advantage and disadvantage. While we make sure that we only accept "good" bids, we also disregard bids that could close to 0,8 and might be on the Pareto frontier for example. Our strategy generally puts more pressure on the opponent to accept. We also fail in the fact that any bid that is above 0,8 is accepted, thereby eliminating the chance of following bids possibly having higher utilities.

3.2 Bidding strategy

For our bidding strategy, we kept our theme of time dependency and added quite a bit of unpredictability. While testing negotiations in genius, we noticed a trend with most agents. They tend to begin their bids with their optimal bids, i.e. where their utility is 1. However, most negotiations don't end with an agreement where an agent has a utility 1. That is to say, most agents end up conceding a bit to reach an agreement. What we wanted to achieve was to confuse the opponent's opponent model by not starting off with our optimal bid. Of course, though we wanted to bid with something that is considered to be quite good.

Our strategy is to begin our bidding at a utility equal to some threshold $y < 1$ so $UA = y$. During the first phase of negotiations, or phase 1, our agent gradually increases its bids so that they peak at utility 1. After they have reached the peak we move on to phase 2 where our agent assumes a conceding strategy.

We decided to stay consistent and have our threshold the same as acceptance strategy threshold, $y = 0,8$. Phase 1 consists of the first third of the negotiation time and phase 2 consists of the rest of time.

The advantages of this strategy are to deceive our opponent during Phase 1. If the opponent should think we start at our optimal bid then it would seem that we are conceding during this Phase. Therefore the opponent may be more inclined to accept a bid although our utility is still close to our optimal utility. Also, we guarantee that if a bid is accepted during this Phase then we should have a utility between 0,8 and 1,0. Another advantage is that we can use Phase 1 to learn about the opponent's bids and create an opponent model.

There are a few disadvantages to this strategy. If we reach an agreement very early then we have eliminated the chance of our agent having an optimal utility. Also, since we are trying to deceive the opponent then we could be lowering the chances of reaching a good agreement for both parties. Another disadvantage is that if the results in Phase 1 are not satisfying then we continue into Phase 2, which is a simple conceding strategy where our utility always decreases.

The following are equations that depict our exact formulas for phases 1 and 2:

$$U = \begin{cases} Pmin + (1 - Pmin)nt, & \text{t} \leq \frac{1}{n} \\ Pmax - (Pmax - Pmin)t^e, & \text{t} > \frac{1}{n} \end{cases}$$

If we have $Pmin = 0.8, Pmax = 1, n = 3, e = 2$:

$$U = \begin{cases} 0.8 + 0.6t, & \text{t} \leq \frac{1}{3} \\ 1 - 0.2t^2, & \text{t} > \frac{1}{3} \end{cases}$$

3.3 Opponent Model

Our opponent model is based on the HardHeaded Frequency Model. Deciding how to improve the model proved difficult. The current implementation has a drawback of becoming less accurate over time. Continuing on from our thoughts of our Bidding Strategy, we decided that during learning, more importance should be put on early bids. We assume that agents (that aren't completely random) will bid either optimal bids or close to optimal bids in the beginning. Therefore when evaluating the preference weights of an opponent, we should assume that early bids are close to their optimal bids.

In line with our time-dependent strategies, we, therefore, decided that updating our opponent model should rely on the time left in a negotiation. This is also helpful our agent has decided early on what it thinks the opponent's preferences are. Bids later on in the negotiation also have less importance and the model is updated less and less.

The implementation of the current Frequency Model looks at the last two bids the opponent makes. If any two issue values are the same, the weights for those issues increase and all issues are then normalized. This is effective when there is no randomness in bids, however, with some slight randomness, it can be inaccurate. Our way to counter this was to increase the number of bids to look at.

We instead chose an arbitrary but logical number of 5 last bids to look at. In those last 5 bids of the opponent, we see if for some issues we see the same value more than once. If this is the case, that issue's preference weight should increase. However, the weights are incremented less than the current Frequency Model. So we end up with a combination of taking a look at the last 2 bids and the last 5 bids of the opponent.

After receiving good feedback of why we chose 5 bids, we realized that a better implementation would be to not have an arbitrary number. Instead, we see which issue in the domain we are running has the most possible values and use that number.

3.4 Opponent Model Strategy

Since our opponent model will be by definition imperfect, using only the calculated opponent utility for an opponent model strategy seems like a bad idea. In order to solve this problem, we want to take into account the Hamming distance of our bids to the last opponent bid (in other words, it is the number of issue values that differs between the two bids). Indeed, the Hamming distance will give us information about how close we are to the opponent bid. The problem is that the Hamming distance alone doesn't take into account the weight of the issues, thus we want to use a combination

of the Hamming distance and the utility estimation of our opponent model.

The new problem that arises is knowing how important the Hamming distance should be compared to the utility estimation. We will use a simple weighted average that can be parameterized by a parameter w. The formula that we are using is the following :

$$bidValue = \frac{w * (1 - Hamming(bid, oldOpponentBid)) + opponentUtility(bid)}{w + 1}$$

In order to find the best bid, we just have to take the bid with the highest bidValue.

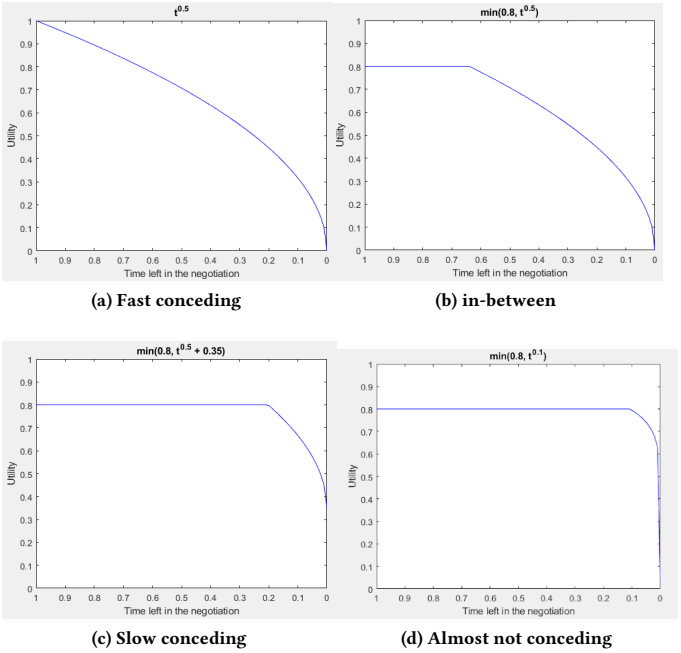
3.5 Strategy under uncertainty

We use similar strategies to that in Opponent Strategy to estimate the utility space under preference uncertainty. The general idea is to traverse from the highest ranking bid down to the lowest ranking bid and add more weights to the issue of which the value remains unchanged while the overall ranking decreases. The implementation is similar. We start traversal from the higher ranking bids, and for each bid, we look back at a number of bids, which is the maximum number of values among all issues in the domain, increase the weight of the issue if any issue value appears more than one. After normalising all weights, we form the estimated utility space. Then combination strategies of estimated utility space and ranking rules are applied to Acceptance Strategy and Bidding Strategy.

For Acceptance Strategy dealing with uncertain profiles, our normal strategy will do just fine. However, to make sure that our agent is able to accept bids with as high utility as possible, we apply an additional rule that checks if the bid ranks at least as high as the acceptance threshold which is a function of a constant and time mentioned above. For example, at some time, our calculated acceptance threshold is 0.8, then a bid with an estimated utility higher than 0.8 and ranks at least 80% in the bid order shall be accepted immediately.

As for Bidding Strategy under uncertainty, we stick to the previous method that bidding towards a utility goal with or without using the OM. However, there is no absolute guaranteed that our estimated utility space is accurate enough to find a bid that matches the utility goal. So we check if the bid found by the utility space does rank at least as high as the percentage value of the utility goal in the bid order. If it does, we offer the bid to the opponent. If not, an arbitrarily small number, i.e. 0.01, is added to the utility goal. Then keep bidding till it's rank is satisfied. For instance, the utility goal at some time is 0.8, if the bid found by the utility space doesn't rank higher than 80% in the bid order, we increase the utility goal to 0.81 and keep bidding.

The combination of estimated utility space and ranking rules performs better than any method alone. Our estimated utility space tends to overestimate utilities with high uncertainty degree, i.e. more than 1500 bid rankings. So it happens that our AS accepts a high-utility bid but actually with a low ranking. This is why we add the ranking rules as a double insurance. After playing in several tournaments against many agents, we observed that the combination strategy does help beat other agents with higher agreement utilities. However, we sometimes stuck in the strategies and result in no agreements.



(a) Fast conceding **(b) in-between**

(c) Slow conceding **(d) Almost not conceding**

Figure 1: Changes in the Acceptance strategy

4 STRONG AND WEAK POINTS

Regarding the strengths and weaknesses of our party, some of them are already defined in the specific section of the components of the agent, however, in this section we summarize them together.

With respect to the acceptance strategy, at the beginning we defined our acceptance as a function of the square root of the time, although this led us to concede excessively quick, ending with low utilities in our side and high utilities for the opponent. We then decided to modify it taking into 0.8 as minimum acceptable utility for the most of the negotiation and then concede much faster at the end of the negotiation to reach an agreement. The evolution of our acceptance strategy can be seen in the figure1.

Our final acceptance strategy still has some weaknesses, such as, we could be missing better deals for us by having the threshold of 0.8 since the beginning, and the negotiations could end quickly if the initial bids are sufficient good for both parties, avoiding us to explore more optimal bids later in the negotiation.

Although, this acceptance strategy guarantee us to obtain a good outcome in the 80% of the time that the negotiation lasts, even if this outcome it is lower than our opponent's outcome.

While developing our agent and making the aforementioned changes, in order to test which acceptance strategy could be the best one for our party we set up a testing situation in which our party would play a tournament against two of the most stubborn agents available, the BoulwareNegotiationParty and the HardHeaded party, in the domain for which this assignment was though, the party

domain, and with the same set of preference profiles in every tournament run. The reason to choose these agents was to test how weak our acceptance strategy could be, so that both agents could be push to their limits and the negotiation could last a high number of rounds, thus evaluating the performance of our agent in advanced stages of a negotiation. We compared the results obtained with the acceptance strategies $min0.8, t^{0.5}$ and $min0.8, t^{0.1}$ so as to decide which of the two we should implement.

From Table 1 we can observe a noticeable improvement in the outcomes that our party is able to get, making also to decrease, thus, the opponent's utility. With the original strategy (#1) out mean utility was below the desired threshold, whereas in the modified one (#2), as it underwent a rise of almost a tenth, the mean utility is above that threshold. The modification also allowed us to reach more efficient agreements, since the distance to the Pareto Optimal Frontier was reduced and also to the Nash solution.

Nevertheless all the improvements, later we realized that this strategy (#2), with which we only concede a little below a 0.8 utility at the final stage of the negotiation, can lead us, occasionally, to a no-agreement outcome, result we want to avoid. This is what happened when playing against the HardHeaded party. Of a total of 216 negotiations, 38 ended with no-agreement, approximately an 18% percent of the total (we can see that in Table 2).

We decided to soften our acceptance strategy in its final stage, by implementing it as $min\{0.8, (t^{0.5}+0, 35)\}$. This change allowed us to reduce the number on negotiations ended with no agreement from 16 to 6 when playing with the HardHeaded agent, which represents approximately a 3% of the total. In the results we observed that, on average, we were even closer to the Nash solution and, then, to the Pareto Optimal Frontier, and our average outcome utility was, indeed, above our defined threshold, which we felt satisfied with.

The biggest strength point of our bidding strategy is that it was thought to deceive our opponent regarding our strategy and our preference profile, since the majority of the opponent models consider that the opponent follows a conceding strategy, starting in its highest utility. This was implemented in this way in order to the opponent not being able to learn our profile and, thus, to not being able to exploit us.

However, this strategy also lead us to prevent a good agreement for both parties if our opponent is trying to make nice movements in his bidding strategy by learning our preference profile. What he would consider a nice movement for us, it could be, actually, a bad movement. Moreover, we could be losing the opportunity to get quickly an agreement with our optimal utility, and, instead, when reaching agreements quickly with our bids, they usually are closer to our threshold than we would like. Nevertheless, we decided to maintain this strategy since it is differentiating from the most common one, which would be starting at optimal utility and, from that, decreasing the target utility in the bids made.

5 CONCLUSION

Working together as a team proved very successful and enjoyable. Our team members come from various places around the world and have different backgrounds. This reflected on our work and resulted in very productive discussions and design choices. Most of us had no prior knowledge of negotiations so we could view the negotiation environment critically and make decisions simply on what we thought best instead of what may be considered traditional methods.

Because of our untraditional methods, our strategy may not apply to all negotiations. Indeed, in many cases during negotiation, we "lose" against the opponent. However, in almost every case we do get a result with some utility and often are very close to the Pareto optimal frontier. Also in many cases our negotiations end up with a result above our threshold of utility 0,8. Therefore, we believe that our agent could be very applicable to negotiations where you are not necessarily interested in getting an optimal result, but would like a result above some value. So if you know before a negotiation that you would like to end up with a utility of at least 0,8, our agent would be a good choice.

The most important environment where our agent excels is in tournaments. In single negotiations we may often "lose" against the opponent but still have a pretty good outcome. That means that in a tournament most negotiations end up with a good enough result for our agent. Other agent negotiations may however end up badly for either agent or even result in no agreement. The following are a few tournaments that show results for our agent.

5.1 Some results

If we test our agent against ourselves, a HardHeaded agent, *theNegotiator* (ANAC 2011) and *Gahboninho* (ANAC 2011) we can see very clearly how our outcomes vary between negotiation sessions and tournaments. The HardHeaded agent was created using the HardHeaded acceptance strategy, bidding strategy and opponent model and the BestBid opponent model strategy. First we shall look at the Negotiation sessions in Table 3.

Here we can clearly see that our average utility is always above 0.7. But our utility is always lower than the opponents utility. Our individual results are not very good because our opponents are quite stubborn and do not concede much. Hardheaded and Gahboninho just keep on bidding their best bids on and on until we lower our acceptance threshold and finally accept their bids. That also explains the quite high number of rounds.

Now let's look at a tournament between these 4 agents, where repetition is allowed (Table 4).

As we can see in the table above our mean utility is the highest and we are closest to the nash product in general. We also come in as a close second in the pareto distance and the welfare. What seems to happen here is that HardHeaded and *Gaboninho* are stubborn to concede. We assume that negotiations between them often lead to no-agreements. However our agent is content with results that are pretty good. So even though we don't do really good in individual negotiation sessions, it seems to be working well to use our strategy in tournaments.

5.2 Other domain

We tried to run tournaments in two other domains as well against the same agents to see the difference between the outcomes. Table 5 and Table 6 see the outcome of these tests. We can see that in the first case we have the highest mean utility, but the second highest in the other one. We are also in the same ranking regarding minimizing

Table 1: Tournament results against BoulwareNegotiationParty in the party domain with the three different acceptance strategies.

Acceptance Strategy	#1$min\{0.8, t^{0.5}\}$		#2$min\{0.8, t^{0.1}\}$		#3$min\{0.8, (t^{0.5} + 0.35)\}$	
Group9 vs Boulware	Group9	Boulware	Group9	Boulware	Group9	Boulware
Total Undiscounted Utility	162.4536	202.6427	182.893	187.6709	182.2433	188.2013
Mean undiscounted	0.7521	0.9381	0.8427	0.8688	0.8437	0.8713
Total Nash Dist	33.4776	33.4776	21.1901	21.1901	21.0022	21.0022
Total Pareto Distance	5.6836	5.6836	4.5229	4.5229	5.6135	5.6135
Mean Nash Distance	0.1549	0.1549	0.0981	0.0981	0.0972	0.0972
Mean Pareto Distance	0.0263	0.0263	0.0209	0.0209	0.02598	0.02598
MeanWelfare	1.6902	1.6902	1.7155	1.7155	1.7150	1.7150
Negotiation Sessions	216		216		216	
Number of session with no-agreement outcome	0		0		0	

Table 2: Tournament results against BoulwareNegotiationParty in the party domain with the three different acceptance strategies

Acceptance Strategy	#1$min\{0.8, t^{0.5}\}$		#2$min\{0.8, t^{0.1}\}$		#3$min\{0.8, (t^{0.5} + 0.35)\}$	
Group9 vs HardHeaded	Group9 $min\{0.8, t^{0.5}\}$	HardHeaded $min\{0.8, t^{0.5}\}$	Group9 $min\{0.8, t^{0.1}\}$ Total / agreements	HardHeaded $min\{0.8, t^{0.1}\}$ Total / agreements	Group9 $min\{0.8, (t^{0.5} + 0.35)\}$ Total / agreements	HardHeaded $min\{0.8, (t^{0.5} + 0.35)\}$ Total / agreements
Total Undiscounted Utility	141,4747	211,6890	139,1098	168,6197	155,7227	201,0462
Mean undiscounted	0,6549	0,9800	0,6440 / 0,7815	0,7806 / 0,9473	0,7209 / 0,7415	0,9308 / 0,9574
Total Nash Dist	55,8633	55,8633	69,7060	69,7060	42,619	42,619
Total Pareto Distance	2,3766	2,3766	43,2798	43,2798	8,88873	8,88873
Mean Nash Distance	0,2586	0,2586	0,3227	0,3227	0,1973	0,1973
Mean Pareto Distance	0,0110	0,0110	0,2003	0,2003	0,0412	0,0412
MeanWelfare	1,6350	1,6350	1,4246	1,4246	1,6517	1,6517
Negotiation Sessions	216		216		216	
Number of session with no-agreement outcome	0		38		6	

Table 3: Outcome of 20 negotiation sessions against each opponent

	Group 9	TheNegotiator	Hardheaded	Gahboninho
number of pareto optimal outcomes	7	13	18	15
average distance to pareto	0.09012	0.007971	0.003836	0.2091
number of Nash products	4	2	1	1
average distance from Nash	0.08593	0.09894	0.1832	0.41304
Group 9 average utility	0.8623	0.8513	0.7245	0.7174
Other party	0.8192	0.9069	0.9659	0.9579
average number of rounds	26.25	101.91	116.2	76.85

the pareto and nash distance. So overall it seems like our agent does a quite nice job in the tournaments in more than the party domain.

5.3 Further Work

There are a few aspects that it would be interesting to change and could result in some improvements. However they would not necessarily improve and would change our overall strategy theme. Nevertheless, changing them and testing heavily would be interesting. The following are changes that would be nice to look at:

(1) Update our acceptance strategy to take into account whether the opponent is conceding. At the moment, our strategy accepts anything that is above our threshold of 0,8 (decreasing with time). This obviously eliminates the possibility of getting better utilities. What could be done is to track whether the opponent's bids are increasing our utility before accepting. Also using the opponent model to see whether we think the opponent is conceding. If these are true, our agent should wait and allow our utility to increase until finally accepting.

Table 4: Outcome of tournament with repetition against the same opponents as in table 1

	Group 9	TheNegotiator	Hardheaded	Gahboninho
Total Undiscounted Utility	79.435	76.137	28.327	31.594
Total Nash Dist	261.302	264.334	371.008	369.858
Total Pareto Distance	188.317	183.196	308.795	305.361
Mean Nash Distance	1.021	1.033	1.449	1.445
Mean Pareto Distance	0.736	0.716	1.206	1.193
Mean undiscounted	0.31	0.297	0.111	0.123
ManWelfare	1.334	1.355	0.33	0.349

Table 5: Outcome of a tournament with repetition against the same opponents as in table 1, but now in the holiday domain.

	Group 9	TheNegotiator	Hardheaded	Gahboninho
Total Undiscounted Utility	163.851	156.617	105.834	99.994
Total Nash Dist	146.749	166.833	291.528	303.501
Total Pareto Distance	108.743	127.256	255.99	267.913
Mean Nash Distance	0.573	0.652	1.139	1.186
Mean Pareto Distance	0.425	0.497	1	1.047
Mean undiscounted	0.64	0.612	0.413	0.391
ManWelfare	2.713	2.561	1.522	1.427

Table 6: Outcome of a tournament with repetition against the same opponents as in table 1, but now in the Bank Robbery domain.

	Group 9	TheNegotiator	Hardheaded	Gahboninho
Total Undiscounted Utility	170.08	175.849	167.329	113.039
Total Nash Dist	127.813	122.192	146.981	276.456
Total Pareto Distance	69.058	66.635	89.654	172.039
Mean Nash Distance	0.499	0.477	0.574	1.08
Mean Pareto Distance	0.27	0.26	0.35	0.672
Mean undiscounted	0.664	0.687	0.654	0.442
ManWelfare	2.781	2.813	2.593	1.599

This could result in higher possible utilities but could also result in missing out on high utilities if the opponent suddenly makes more selfish moves, decreasing our utility.

(2) At the moment our agent is very deterministic. Generally adding some randomness in our bidding strategy could make our agent more successful. This could put off the opponent's opponent model of us and make our bidding more probabilistic. Again though, this may not improve our strategy at all but might still be nice to do.

(3) Finally we could update all constants and factors in all models. All of them are currently chosen logically but we can't prove that they are optimal. Also different constants may be better for different domains. Genius proves difficult with testing easily, but with better testing we could test extensively and compare what constants work well with what domains and how we could update them for better results.

REFERENCES

[1] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. 2014. Genius: An Integrated Environment for Supporting the Design of Generic Automated Negotiators. *Computational Intelligence* 30, 1 (2014), 48–70. https://doi.org/10.1111/j.1467-8640.2012.00463.x