The SliBwaTer Agent

Sam Leurink
Marc Overbeek
Sacha Vucinec
Pieter van der Werff

ABSTRACT

Abstract In this report we present our agent. It combines various acceptance strategies with a complex bidding strategy and modelling our opponents reservation value. The goal of this strategy is to create a hard-headed agent, that will still get good results in a tournament.

1 INTRODUCTION

This work briefly outlines our SliBwaTer agent, which was designed for the Automated Negotiations League (ANL) of the Automated Negotiating Agent Competition in 2024 (ANAC 2024). In this competition, two agents enter a negotiation conducted in a domain with several different features. Each agent is assigned an arbitrary configuration of preferences for feature values. The goal of the agents is to maximise the utility they gain from the negotiation by trying to reach an outcome that is beneficial for themselves, but also feasible enough for the other agent to come to an agreement.

2 NEGOTIATION STRATEGY

Our SliBwaTer agent is characterised as an agent that will offer an outcome that offer high utility and will concede to less beneficial outcomes, following a dynamic concession rate that is dependent on the relative time in the negotiation and the concessions of the opponent. If the opponent makes an offer, the SliBwaTer is not keen to accept it unless the offer is better than the outcome the SliBwaTer itself would have offered in the next turn. Towards the end of the negotiation, our agent will be more lenient towards accepting incoming offers.

In order to estimate the negotiating position of its opponent, the SliBwaTer agent will try to construct a model of the opponent, based on its behaviour. In the ANAC 2024, an agent has perfect knowledge of the function that determines the opponent's outcome feature preferences: its utility function. This knowledge of its opponent's utility function gives us great knowledge about the utility distribution in the domain outcome space. However, an agent does not know the opponent's reservation value: the utility that an agent gains if no agreement is reached. As a lower reservation value will increase the need to come to an agreement with the opponent, this hidden property of an agent is likely to be reflected in its behaviour.

To model the opponent's reservation value, SliBwaTer analyses the opponent's order history and uses regression to form an estimate, based on a method used by Chao Yu et al [9]. How exactly this is done will be thoroughly described in section 2c.

Since our agent has full access to the opponent's utility score, we have more information to include in the behaviour of SliBwaTer. Where an agent would typically mostly base its offers on its own utility, it can now also take its opponent's utility into account with great precision. Therefore, concessions can also be measured in terms of raising the opponent's utility, rather than only lowering the agent's own utility. To exploit this knowledge for the benefit of SliBwaTer's position in the negotiations, we experimented with a measure of an outcome's utility score, taking into account both the agent's own utility and its opponent's. This alternative calculated utility score will be discussed in more detail in section 4.

The SliBwaTer agent adheres to a BOA agent architecture [1]. For the remainder of this section, the individual components of the agent will be discussed in more detail.

2a Acceptance strategy

The Acceptance strategy uses a combination of different acceptance and rejecting conditions.

- Always reject condition. The always reject condition is the first condition the agent checks. This condition checks if the offer we got from our opponent results in a value that is below our reservation value, plus half our Mu value. Our mu value is one of our hyper parameters that indicates how much we fake our reservation value being higher then it actually is. This way we cannot accept an offer that is worse then the reservation value, while also making sure that any offer we accept is at least slightly more beneficial then just not coming to an agreement at all. This condition is the first one because for a large part of the negotiation any offer made by our opponent wont be sufficient, and therefore saves on time by not needing to check with the other conditions.
- Always accept condition. This condition always accepts the last offer the opponent can send. This is just to be sure that we don't miss out on any offer that would be more beneficial then not coming to an agreement at all. All the offers that are not beneficial will already have been rejected by the always reject condition. Therefore it is always good to accept any offer that we get on the last round.
- Better then next offer. If the offer we get is more beneficial for us compared to our opponent then the next offer we were going to send to our opponent, we will simply accept the offer. This condition takes our utility function into account, but also takes our opponents gain into account. For this condition to succeed the offer has to be both better for us and we need to gain more then our opponents does. This is because sometimes we want to send a worse offer, making the opponents offer better then our next, if we only look at our own utility function.
- Final stretch condition. [2] This condition only activates at the final 10 % of the negotiation. It checks how many steps are left in the negotiation, and then looks that amount of steps back in the offer history. If the current offer is the best offer out of all of those, Then it will accept the offer. This

makes sure that we get the best offer we can get at the end of the negotiation. The further into the final stretch we get, the smaller the window will be that we look back at. If there are 5 rounds left we will only check if the current offer is better then the 5 previous offers. This way the condition for acceptance gets broader as we get closer to the deadline.
- If none of the conditions apply to the current offer, we will reject the offer.

By using the combination of accept/reject conditions, we believe that we only accept beneficial offers, while also balancing pushing for better offers with the time pressure of the end of the negotiation.

2b Bidding strategy

The most important part of the agent is the bidding strategy. The bidding strategy decides the concessions we make over time when we decide not to accept the opponent's offer. This could be done in a straightforward way (e.g. linear or random agents), but the agent could profit greatly from using more complex functions to choose a bid. For this, we have split our strategy into a multiple parts parts.

Concession Curve: The concession curve is possibly the most important part of the strategy. Many agents such as Boulware [7] and Conceder [6] use at least some form of concession curve to calculate their bids. Since we aren't sure which exact one is the best at what it does, we decided to take the function as described by Fatima et al. [4] which formulates most concession curve as a standard function with a variable. This function is:
$\phi^a(t) = k(a) + (1 - k(a))(\frac{min(t,T^a)}{T^a})^{\frac{1}{\psi}}$
Here $k(a)$ is a constant, t is time and and T^a is the total time in the negotiation. Since $k(a)$ is a constant that isn't defined yet, we can interchange this with our reservation value rv^a. The final adjustment we made to get the correct curve is inverse the relative time to make the curve go down instead of up. The final function therefore looks like this:
$\phi^a(t) = rv^a + (1 - rv^a)(1 - \frac{min(t,T^a)}{T^a})^{\frac{1}{\psi}}$

Concession Limiting: Having a standard concession curve is nice, but doesn't always allow for the best outcomes. In the past agents such as HardHeaded [8] and Gahboninho [3] have done well by adopting a "No quarters given" strategy which only starts to concede at the end of a period. With a ψ close to 0 this shouldn't be too much of a problem, as the concession rate itself will be really low at the start anyways, but for the values of ψ closer to 1, it is more important to help prevent a situation where our utility is already significantly lower than theirs by the time they start conceding. AgentK2 [5] have already implemented a concession limiter of sorts, but have a different function than we desire as it takes opponent utility function modelling and confidence into account. For us this is of course unnecessary thus we decided to create a basic function which is: $r(t) = \frac{o}{c*(1-t)}$
In this case o is the opponent's concession and c is our own concession rate. To prevent us conceding more than we originally planned and negatively if the opponent falls back on concession we have the function:

$$r(t) = \begin{cases} 1 & \frac{o}{c*(1-t)} \geq 1 \\ \frac{o}{c*(1-t)} & 0 \leq \frac{o}{c*(1-t)} \leq 1 \\ 0 & otherwise \end{cases}$$

Along with that if $c = 0$ or $t = 1$ we return 1 as $r(t)$

Artificial Reservation Value: The current negotiation environment is very transparent, knowing both your own and opponent utility. This does leave one unknown: the opponent's reservation value. Of course we will look at the opponent's reservation value in opponent modelling, but the way our opponent sees ours is also important. To make sure we look better than we actually are we will create an artificial reservation value called rv^a. For this we want to use our modelled opponent reservation value but also a fail safe to make sure we always look stronger than we actually are. To calculate the correct rv^a we therefore have split it into two functions:

(1) $rv^a = rv + \mu$.
We will have a standard reservation value which takes the original reservation value rv and adds parameter μ to it.
(2) $rv^a = pf(orv + \rho)$
This function implements the modelled opponent reservation value: orv to calculate the reservation value we can pretend to have. This along with the confidence rate ρ gives the reservation value we expect the opponent to have. To transfer this into a reservation value of our own we execute the function $pf(rv)$. This function isn't a mathematical function, but a function that checks the Pareto frontier for the closest point to the opponent's reservation and returns the own reservation corresponding to it.

Finalized concession. Combining these parts for our final concession will give the function:
$c(t) = (\phi^a(t-1) - \phi^a(t)) * c$
In this case ϕ is calculated with the artificial reservation value rv^a

2c Opponent modelling

A good model of opposing agents requires an estimation of the strategies and reservation value of the opponent, based on its behaviour. In order to come up with a good modeling strategy, we first identified likely potential opponents.

Types of agents. We can subdivide our possible opponents into a few categories. It should be noted that the utility function of our opponent is known to our agent, so we can use this to our advantage when predicting the opponent's next moves.

- **Boulware-like agent**: this agent will lower their minimum gained utility over time, using a predetermined function of time and its reservation value. This opponent's strategy isn't notably influenced by our own agents' behaviour. Depending on the complexity of this opponent's predetermined function, we could form an estimate of the opponent's future bids, as well as their reservation value. We could use a bilateral negotiation model [9] to estimate our opponent's bidding curve and reservation value.

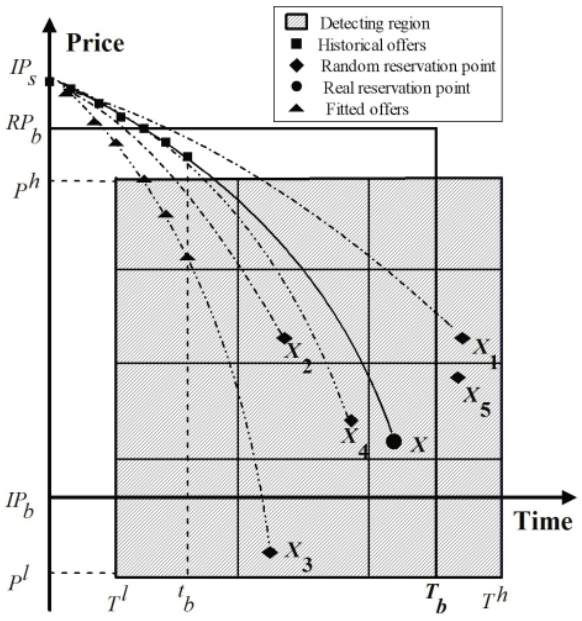


Figure 1: Detecting region used for opponent modelling. Source: [9].

- **Tit-for-tat-like agent**: this opponent will actively look at (the utility of) our agents' bids, and will only concede after our agent has conceded first. The tit-for-tat opponent usually also looks at how much we have conceded, and then decides how much it will concede based on that (for simplicity, we will assume this to be a constant factor n of our agents' concession). A smart opponent will wait for our agent to stop conceding before it itself starts to concede. We believe this agent's reservation value is difficult to predict. However, if we suspect our opponent to be of this type, we could form an estimate of the factor n by doing some small concessions early on to see how our opponent reacts, after which we could increase our minimum utility again.
- **Hybrid agent**: the most complex and the most difficult type of agent to face. This agent does not have a predetermined strategy, instead it bases its strategy on the behaviour of its opponent to try to achieve the best outcome for itself.

Reservation value estimation. In order to estimate our opponent's reservation value, we use a method based on what is described in [9]. We form a *detecting region* (see figure 1) with several sub-regions. The detecting region spans from T^l to T^h in the time dimension, and from P^l to P^h in the price dimension. Only reservation values in this area will be modelled. In our implementation, $(P^l, P^h) = (0, 1)$, according to the tournament settings. The final deadline is already known to both agents, so $T^l = T^h =$ `self.ami.n_steps`. We will call this deadline value d. We set the number of detecting sub-regions N to 100. In [9], it is shown that the overall detection improves as N increases.

At each round t_b, an estimate of the opponent's reservation value will be formed as follows:

(1) For each detecting sub-region C_i with price domain $[P_i^l, P_i^h)$, select a random reservation value r_i in this price domain.
(2) For each r_i, calculate a fitted function (dashed lines in 1) based on the opponent's order history H. We use the following function:

$$Offer_i(t) = p_0 + (r_i - p_0)\left(\frac{t}{d}^b\right),$$

where t is the time step, p_n is the n-th price in the order history H and b is defined as follows:

$$b = \frac{\sum_{j=t_1}^{t_b} t_j^* p_j^*}{\sum_{j=t_1}^{t_b} t_j^{*2}},$$

where t_1 is the time step of the second[1] round saved in the order history H. Also, $t_j^* = \ln \frac{t_j}{d}$ and $p_j^* = \ln \frac{p_0 - p_j}{p_0 - r_i}$.
(3) We compare all entries (t_j, p_j) in the order history to their fitted offers. We calculate the mean squared error e_i for each C_i.
(4) The sub-region C_i^* with the lowest e_i is selected to contain the estimated reservation value of the opponent. The estimated reservation value will be set to r_i.

The order history H is the bidding curve of the opponent, however, there is a small difference. The order history H starts at the highest bid of the opponent (measured in opponent's utility). This is done for the reasons listed below.

- We now always have a regression curve that is decreasing over time. An increasing curve wouldn't make sense, because for that, the opponent would need to bid below its reservation value.
- We possibly sidestep some of our opponent's 'tricks' to confuse our opponent modelling function. The opponent's bids could for example repeatedly go up and down.
- When an opponent keeps sending offers with the same utility at the start of the negotiation, the opponent modelling system will disregard these offers, except the last one with this utility. This works well against hard-headed or boulware-like opponents who will only concede at the very end of the negotiation.

It is important to note that the reservation value estimate is only accurate when the opponent has started conceding. Before this moment, the reservation value will be significantly overvalued.

An example of the opponent modelling system estimating the opponent's reservation value can be seen in figure 2.

3 AGENT PERFORMANCE

3a Tests run

There are several hyperparameters to be fine-tuned. We looked at the following three hyperparameters:

- ψ (range $[0, 1]$): this parameter influences the shape of our agent's bidding curve. A lower value of ψ will lead to a steeper bidding curve at the end of the negotiation, while a

[1]The second round is used, so we do not divide by zero.

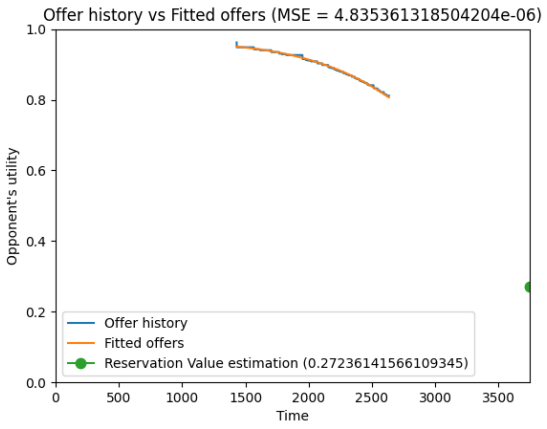


Figure 2: Opponent's fitted offers compared to their actual order history when playing against a boulware agent. You can see how the opponent modelling system disregards the first part of the negotiation.

higher value of ψ spreads out the agent's concessions over a longer time period. We looked at the following values of ψ in our tests: $\Psi = \{0.05, 0.1, 0.2, 0.5\}$.

- μ (range $[0, 1]$): minimum added value to our actual reservation value, to calculate the artificial reservation value. See section 2b. $\mu = 0$ will lead to a normal boulware agent that eventually concedes towards its reservation value. A value of 1 will lead to such a high artificial reservation value that or agent will never concede. The following values were tested: $M = \{0.0, 0.1, 0.2\}$.
- ρ (range $[0, 1]$): confidence rate of the opponent's reservation value, estimated by the opponent modelling system. Lower values mean increased confidence. The following values were tested: $P = \{0.0, 0.05\}$.

The different combinations that were used in testing are shown in table 1. In each test, we ran a tournament with five repetitions against the six default ANL agents: Boulware, Linear, Conceder, RVFitter, NashSeeker and MiCRO. The results of our experiments are discussed in the next subsection.

3b Results

On average the agent did well. In figure 3 you can see that SliBwaTer is close to the Boulware agent when it comes to the average score, even edging out the Boulware agent in the best case. Because we have tested so many different configurations the data for this could be skewed though, as some will have done better than others. Therefore it is good to take a closer look at what value for each parameter gives the best outcome and if the parameter has any impact on the result whatsoever.

Psi. Psi was the parameter we expected to have the most impact on result. Based on literature of past competitions, concession curve's who show a steeper concession near the end of the negotiation tend to do better than agents that take a more linear approach to their reservation. The opposite is shown in our case when you take a look at figure 4a. Here you can see that both the average score and the outliers for $\psi = 0.05$ is noticeably low in comparison to other values of ψ. This is disproved though if we take a look at figure 4b, which plots the scores based on difference between our agent and the Boulware agent. Here you can see that on average the lower values of ψ do better than the higher values of *psi*

ψ	μ	ρ
0.05	0.0	0.05
0.05	0.1	0.05
0.05	0.1	0.0
0.05	0.2	0.0
0.1	0.0	0.05
0.1	0.1	0.05
0.1	0.1	0.0
0.1	0.2	0.0
0.2	0.0	0.05
0.2	0.1	0.05
0.2	0.1	0.0
0.2	0.2	0.0
0.5	0.0	0.05
0.5	0.1	0.05
0.5	0.1	0.0
0.5	0.2	0.0

Table 1: Different parameter configurations

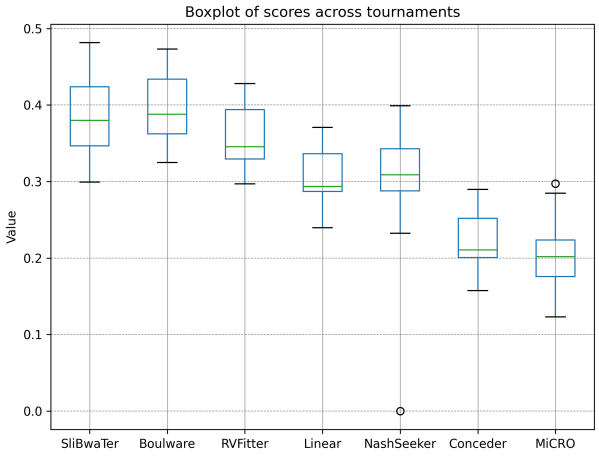
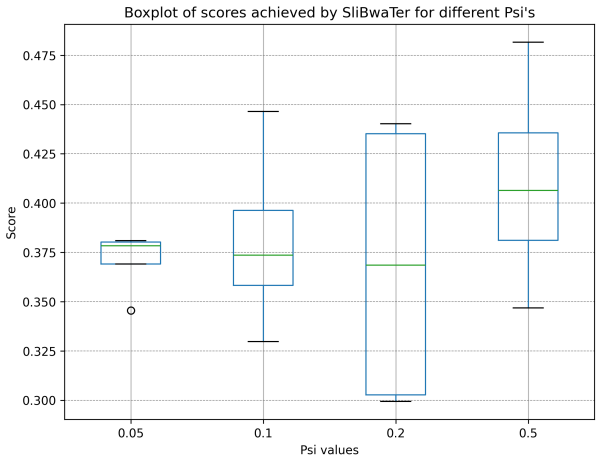


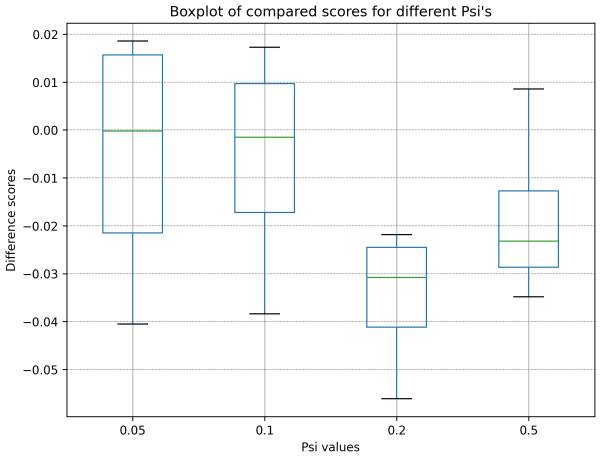
Figure 3: Scores of our agent compared to the standard ANAC agents

Mu. Mu was a parameter we expected to have a lot of impact in our results as well, since adding to our reservation should result in higher utility. Figure 5a shows the opposite. On average the utility goes down the higher the μ, which is a sign that more negotiations failed if we raise the μ. Figure 5a confirms this thought as both average distance and best distance seem to be better than or equal to the other values of μ.

Rho. Rho was a parameter we expected to have very minimal impact on our outcome. This is somewhat shown in figure 6a. As

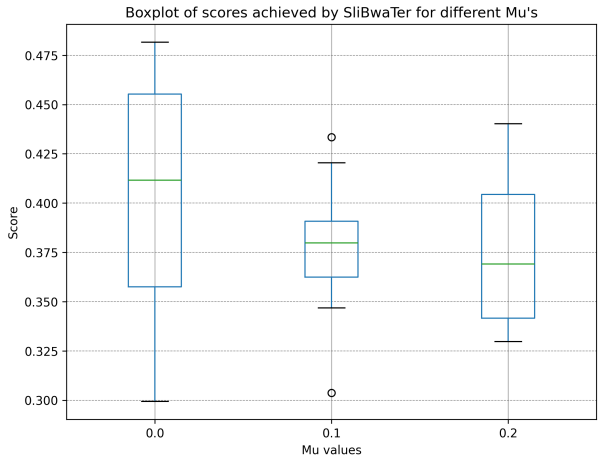


(a) Scores achieved for each psi

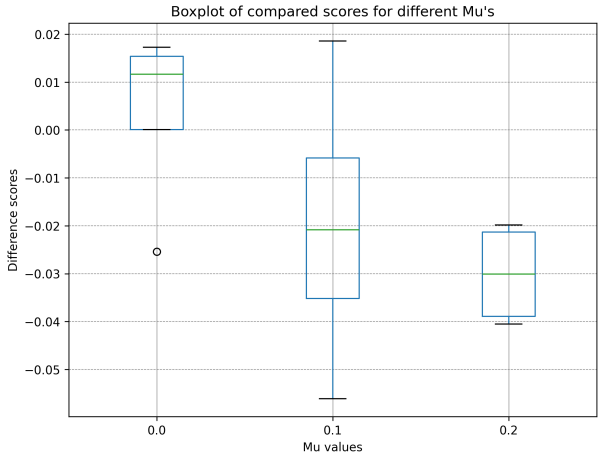


(b) Difference with Boulware agent

Figure 4: Graph a plots the score of our agent in each tournament against the different Psi values. Graph b plots our score compared against the Boulware agent against our different Psi values.



(a) Scores achieved for each mu



(b) Difference with Boulware agent

Figure 5: The lower mu is preferred over the higher mu. In graph a it is shown that both the mean utility and highest peak for mu=0.0 is the highest and in graph b it is shown that it is the only mu with a positive average difference with the Boulware agent

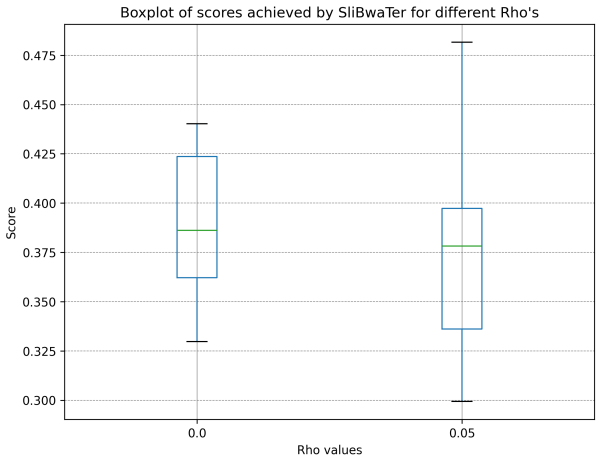
you can see the average scores for both values of ρ are very close to each other, with the higher value of ρ having more outliers than the lower. In figure 6b though we find that the higher value of ρ results in a better respective score.

4 FUTURE PERSPECTIVES

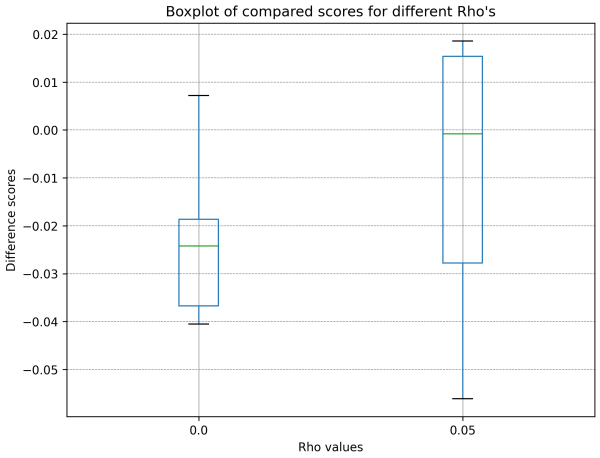
Although we did not achieve any scores that were spectacular, it doesn't mean that there is nothing to be excited about. The function for the bidding curve that we have created still has a lot more values that need to be tested. For Mu we only had 3 values and for Rho we had even less with two. These values can be tested more thoroughly in the future and it maybe even allows for more effective versions of our agent in the future.

Another interesting thing to look at for the future is implementing different utility functions. In our code, we made sure to use a utility score instead of the standard utility function. While the utility score was set to be the same as the agent's utility function the test negotiations, the use of this score allows for alternate utility functions to be created. These functions could include the opponent's utility function, as well as the agent's own utility function (as mentioned in section 2) in such a way that a decrease in the opponent's utility can be valued as an increase in the utility score. We devised such a function and some preliminary testing. The use of this function as the utility score showed promise in some configurations of the SliBwaTer agent, but was too brittle to use effectively in the tournaments. After more research, finetuning and testing, this approach could be interesting for future agent implementations.

Of course there are also the limitations of the negotiation itself. The fact that walking away from a negotiation is often negative



(a) Scores achieved for each rho



(b) Difference with Boulware agent

Figure 6: Graph a plots the score of our agent in the different tournaments against the different Rho values. Graph b plots the difference between our agent and the Boulware agent against the different Rho values.

for both parties means that most of the concessions will be done in the latter part of the negotiation. In a real world scenario there isn't a hard "end of negotiation" limit that we can abuse to get our opponent to give us a better deal. Another thing is that currently we only have two options after each offer our opponent gave us. it might be more useful if we could express our willingness to continue the negotiation. Currently walking away from a negotiation is never a good idea, but if we could threaten to walk away, it might help with getting more hard-headed opponents to concede a bit.

Another idea is that we could keep information between negotiations. If we already have certain information about an agent from a previous negotiation, we could use this information to adjust our strategy to it. This also plays into the threatening to walk away idea, since if an agent walked in a previous negotiation, we might have to be more willing to concede in the next one.

Lastly, the parameter Mu has not only values that can be explored further, but also the possibility to expand that into lambda functions. Having the mu be set to a specific value is nice for simplicity, but could lead to situations where an agreement could never be achieved as the artificial reservation value is too high. This could be prevented by taking the original reservation value into account when calculating the artificial reservation value.

5 CONCLUSION

In conclusion, the SliBwaTer agent is an amalgam of different acceptance, bidding and opponent modeling strategies. While this combined approach did not always outperform other standard ANAC agents, the SliBwaTer agent performs well enough. With some additional research, finetuning and testing, this agent architecture might be further improved to the level of a smart, hardheaded negotiating agent.

Team experience *- by group coordinator*

In general, We have had a very pleasant experience working on the SliBwaTer agent as a team. The group meetings were creative, constructive and productive, and we arrived at a task division that seemed to work well for every member of the team. Unfortunately, our team member Ali dropped out of the course due to unforeseen personal circumstances. However, the rest of the team worked hard together to get the agent working and finish the report on time.

Task division. While we all felt responsible for every part of the agent and the report, it was convenient to appoint a chief to every BOA component of the agent. Sam was in charge of the bidding strategy, and worked out most of the agent performance results. Sacha was responsible for the Opponent modelling, while Marc concerned himself mostly with the acceptance strategy. Pieter, as group coordinator, was responsible for the logistics of the cooperation, as well as being a jack-of-all-trades to be used whenever necessary. The testing of the agent and writing of the report was a pleasant collaborative effort.

REFERENCES

[1] Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alexander Dirkzwager, and Catholijn Jonker. 2014. *Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies.* Springer Japan, Tokyo, 61–83. https://doi.org/10.1007/978-4-431-54758-7_4

[2] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. 2014. Effective acceptance conditions in real-time automated negotiation. *Decision Support Systems* 60 (2014), 68–77. https://doi.org/10.1016/j.dss.2013.05.021 Automated Negotiation Technologies and their Applications.

[3] Mai Ben Adar, Nadav Sofy, and Avshalom Elimelech. 2013. *Gahboninho: Strategy for Balancing Pressure and Compromise in Automated Negotiation.* Springer Berlin Heidelberg, Berlin, Heidelberg, 205–208. https://doi.org/10.1007/978-3-642-30737-9_13

[4] Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. 2002. Multi-issue negotiation under time constraints. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1.* 143–150.

[5] Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. 2013. Agentk2: Compromising strategy based on estimated maximum utility for automated negotiating agents. *Complex Automated Negotiations: Theories, Models, and Software Competitions* (2013), 235–241.

[6] Dean G Pruitt. 2013. *Negotiation behavior.* Academic Press.

[7] Howard Raiffa. 1982. *The art and science of negotiation.* Harvard University Press.

[8] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. 2013. *HardHeaded.* Springer Berlin Heidelberg, Berlin, Heidelberg, 223–227. https://doi.org/10.1007/978-3-642-30737-9_17

[9] Chao Yu, Fenghui Ren, and Minjie Zhang. 2013. An Adaptive Bilateral Negotiation Model Based on Bayesian Learning. In *Complex Automated Negotiations*. https://doi.org/10.1007/978-3-642-30737-9_5