ANL 2021 – "AlphaBIU Agent"

1. Introduction

The agent described below was built for ANL (Automated Negotiation` League).

The agent can elicit his preferences during the negotiation, by learning the opponent's model and negotiate in order to get the bid with the best utility.

Leading concepts for the algorithm:

- Using expended frequency model to estimate opponent utility.
- Compromise after the opponent agent, using learning to model the opponent.
- Use adaptive thresholds during the negotiation to select "good" bids to offer.

2. Agent Model

In automated-negotiation terms, it is customary to represent an agent by BOA model [1]:

1. **B**idding strategy
2. **O**pponent model
3. **A**cceptance strategy

We will discuss each part separately. The first subsection describes the bidding mechanism of the agent. The second subsection describes the opponent model, includes the opponent threshold model and utility function model. The third subsection describe our acceptance methodology and the *adaptive-alpha algorithm*.

2.1. Bidding Strategy

In general, we don't try to offer the optimal bid at every step (when "optimal bid" is used to describe "the best bid we can offer"), but the first bid its value is above the threshold (see section 2.3).

More precisely, we divide the negotiation into two phases, defined by $tPhase$, a hyperparameter that set the time (relative to negotiation duration) that the phase transition occurred. $tPhase$ was set to 0.2 (i.e., 12 seconds for 60 second negotiation). It splits the negotiation into the next phases:

1. 1st phase – in this phase we evaluate the potential bids by our own utility, compared to the utility threshold, and offer the first bid that its utility is above the threshold.

2. 2nd phase – in this phase we evaluate the potential bids by our own utility **and** the opponent utility, as modeled by the opponent model (see section 2.2). The first bid that above our threshold **and** opponent threshold, is offered.

At every turn, we evaluate up to $|X|$ potential bids. X is hyperparameter and was set to 1K in our agent. After evaluating $|X|$ potential bids, if no bid is "good" (as defined above), we offer the *optimalBid* (that is the bid with the highest utility for our agent).

The *optimalBid* is calculated during the initialization phase (the highest-utility bid that found during the initialization phase, up to 50K bids evaluated).

Offering the optimal bid as default offer has three main advantages:

1. At every step we have to make an offer; we didn't find a good offer, but *optimalBid* is good, at least for us ("better than random").
2. The opponent threshold apparently decreases over time – the opponent may accept our offer because its threshold decreased.
3. An opponent that uses frequency-model to model its opponent (that is our agent) updates its model based on our offers. Offering the *optimalBid* helps it to model us better and offer more relevant offers in the rest of the negotiation.

Very close to the end of the negotiation, we evaluate the best bid offered by the opponent. If it's above our threshold (that decreases over time), we offer it, assuming that if the opponent offered it earlier it apparently accepts it now, and late agreement is much better than no agreement at all.

2.2. Opponent model

Opponent model consists of two parts:

- Model of opponent's utility function – its issue weights and values value.
- Model of opponent's threshold (in general, over time).

We used extended frequency model to model the first part, and learning (during learning phases) to model the second.

Frequency model [6] assumes that the mechanism of creating offers is random-based, and similar to our mechanism (evaluating large amount of offers, offer the first that above the threshold). Under this assumption, values with higher value tend to appear more times in the offers. In [3], it was proven to be the most accurate opponent model.

The usual frequency model estimates only opponent **values**, but does not estimate opponent **issues weights**. We extended this model to approximate opponent's issue weights, in order to approximate opponent's utility function.

We used the **standard deviation** of the values frequencies to approximate the **issues weights**. Careful analysis shows that the standard deviation of the frequencies, assuming random-with-threshold offers mechanism, depends on the standard deviation of the real opponent's issue-values (e.g., in a specific issue, issue-values are $v_{1\dots n}$. The higher std(V), the higher std(f) for V as issue-values vector and F as the frequency corresponds to each value.

The other factor affects std(f) is the issue-weight. The higher weight for the issue, the higher standard deviation of the frequencies of this issue. That is because for issue with weight of 0.01, the opponent doesn't care if it gets 0.0098 (issue-value = 0.98) or 0.0023 (issue-value = 0.23) for this issue, but for issue with weight of 0.8, the opponent **does** care if it gets 0.72 (0.9) or 0.64 (0.8).

We assumed that the std-s ratio represents the issue-weights more than the internal distribution of issue-values of each issue. Therefore, we used these formulae to estimate to opponent's utility:

$$v_j^{s_i} = \frac{f\left(v_j^{s_i}\right)}{\max_{\mathrm{k}} f\left(v_k^{s_i}\right)} \qquad s_i = \frac{\mathrm{std}(V_i)}{\sum_k \mathrm{std}(V_k) + \epsilon}$$

Then, the opponent's utility is $\Sigma_i s_i\left(\Sigma_j v_j^{s_i}\delta_{ij}\right)$, for $\delta_{ij} = 1$ iff value j corresponds to issue i in the bid and $\delta_{ij} = 0$ otherwise. We tested our model on our own agent:

Calc. Weight	Real Weight	Issue	Issue-Value	Frequency	Calc. Value	Real Value
0.051939	0.095	Food	Finger	99	0.761538	0.52
			Chips	87	0.669231	0.24
			Handmade	130	1	0.71
			Catering	129	0.992308	0.97
0.271742	0.228	Drink	Non-Alc	74	0.25784	0.33
			Beer	58	0.202091	0.22
			Handmade	287	1	1
			Catering	52	0.181185	0.11
0.098114	0.109	Location	Tent	83	0.198091	0.3
			Dorm	102	0.243437	0.23
			Room	163	0.389021	0.99
			Ballroom	71	0.169451	0.09
0.206453	0.214	Invitations	Plain	173	0.371245	0.84
			Photo	45	0.096567	0.24
			Handmade	41	0.087983	0.12
			Printed	207	0.444206	0.98

0.085185	0.108	Music	MP3	188	0.426304	0.94
			DJ	114	0.258503	0.32
			Band	139	0.315193	0.64
0.286567	0.246	Cleanup	Water	294	1	1
			Materials	34	0.115646	0.36
			Equipment	81	0.27551	0.12
			Hired	60	0.204082	0.21

The normalized relative error of the issue weights if 0.013, that means that the model approximates the opponent's utility relatively good. In practice, we saw that the performance of the opponent's model depends on the utility profile of the opponent. In particular –

- In some domains, the accuracy of the approximation function is better than other.
- But, in all domains, the approximation function captures the opponents' **behavior**, e.g., the function found a linear decreasing (over time) pattern for a linear-decreasing opponents.

In 2.1 we mentioned that we divide the negotiation into two phases. Now we can explain that the reason for this division is the need to stabilize the frequency table (for example, after one offer only, it worth nothing).

After calculating the opponent's utility, we need to estimate its threshold. As default, we assumed for all the opponents constant threshold (about 0.6). We estimated that most of the agents have higher threshold, but we added margin because of the inaccuracy of the opponent model.

We took the advantage of **learning rounds** to better estimate the opponent threshold. For doing this, we divided the 2nd phase into $tSplit$ timeslots ($tSplit$ is hyperparameter, set to 40 in our agent). During the negotiation, for each offer we got, we saved it in the corresponds timeslot. When then negotiation ends, we calculate the average of all the offers in a specific timeslot and remember that value (data from some rounds in weighted, so that later round is more important).

When we meet a known opponent (an opponent with learned data), we smooth the learned data (using moving average with specific window size), and then for each timeslot we compare the utility of the opponent (as calculated by the extended frequency model) to the learned threshold that corresponds to the current timeslot.

The rational of this method is that for an agent with random-and-threshold mechanism to create offers, and bids that distributed roughly-uniform in the utility-space, the average of many offers in timeslot that is wide enough, represents the threshold – for example, if the threshold is 0.9, we expect (under the assumptions mentioned above) to average of 0.95, but if the threshold is 0.7, we expect to average of 0.85.

This rule is symmetrical, therefore when we see average of 0.85, we assume threshold of 0.7.

2.3. Acceptance Strategy

Opponent offers are accepted if the utility is above our threshold. In the same way, bid is considered to be "good" if its utility is above the threshold. Our threshold is changing over time, and the accurate threshold function is determined during the **learning rounds**, for every opponent separately.

Our threshold function (over time) is –

$$\mathrm{T}(t) = V_{max} - \left(\frac{e^{\alpha t} - 1}{e^{\alpha} - 1}\right)\Delta V$$

This function is a monotonic-decreasing function, so that $\mathrm{T}(0) = V_{max}$, and $\mathrm{T}(1) = V_{max} - \Delta V$. We set $V_{max} = 0.95$, the threshold we try to achieve, and $V_{min} = 0.6 \times \overline{V_T} + 0.4 \times \overline{V_O} - \mathrm{std}(V_T)$, for $\overline{V_T}$ that represents our average utility till now, $\overline{V_O}$ is the average utility against the specific opponent, and $\mathrm{std}(V_T)$ is the standard deviation of our utilities till now. $\Delta V = V_{max} - V_{min}$.

The key parameter of this function is **alpha**, which is calculated during learning rounds against each opponent. Alpha defines the **time of compromising**. For example, here is a graph for some different values of alpha:

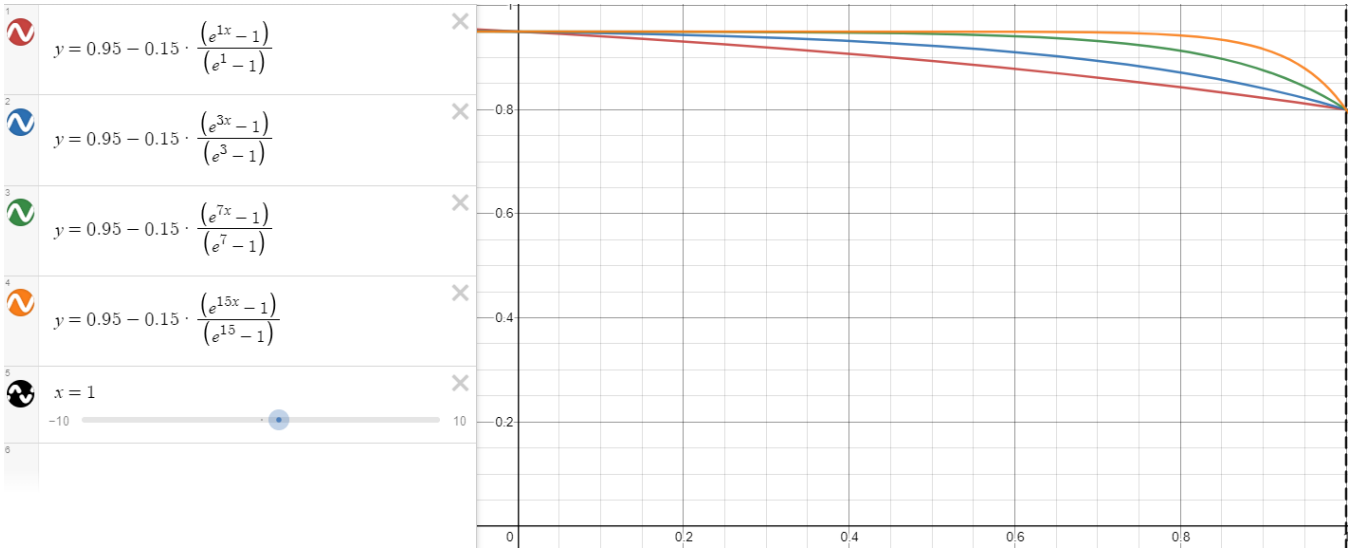

Figure 1 - AlphaBIu threshold function depends on alpha value

As can be seen, the higher value of **alpha**, the later the agent start to compromise. The purpose of alpha is to **start compromising after the opponent**.

For calculating alpha, we're looking for the point that the opponent agent is already compromising, using the array we created for modeling the opponent (see 2.2). For illustration, we show the calculation on example (small) array:

0.91	0.90	0.91	0.88	0.88	0.87	0.84	0.82	0.77	0.76	0.72	0.68	0.64	0.00	0.00

First, the agent extract the initial opponent's utility (array[0]), and the last valid opponent's utility (squares with 0's represents no data, apparently because agreement achieved or opponent left the negotiation before its end). Then, the agent looks for the point that its value is more than 80% of the way from the highest value to the lowest value.

In the example array, this value is $0.91 - 0.8 \cdot (0.91 - 0.64) = 0.69$, so the first index is 11. Each index corresponds to timeslot. In our example, the timeslot is $0.2 + 0.8 \cdot (11/15) = 0.78$, so we want to start compromising (our function decreases by 20% relative to V_{max}).

For doing that we have to extract α, but this is hard equation, and regular math is not enough to extract α. Therefore, we used **polynomial approximation**, by testing some desired values of α and approximate it using 4-th degree polynomial equation:

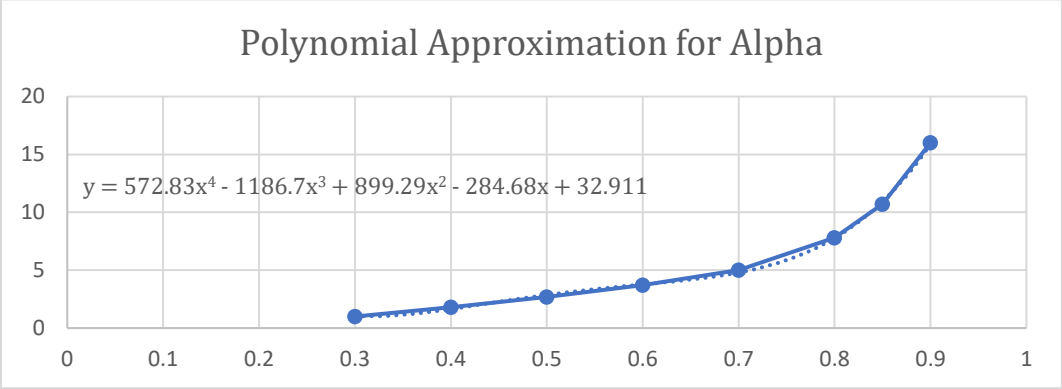

Figure 2 - Polynomial approximation of alpha

This way, we increase our chance to achieve better utility from the negotiation round (because the opponent may compromise before we will), and yet compromise down to $V_{max} - \Delta V$ in the end of the negotiation round, for maximize chance to finish with good agreement.

As mentioned in 2.1, the threshold function is used also for determining if a specific bid is good.

3. Preliminary results

We ran our agent against the sample agents (made by the competition organizers). The table below show average results over many tournaments (though the results are surprisingly stable between different tournaments):

Agent	Average Utility
AlphaBIU (our agent)	0.680
Boulware	0.646
Linear	0.614
Conceder	0.601
Random Party	0.563
Hardliner	0.489

Table 1 - Average utility in tournaments

From the results, we can see that our agent gets high utility against all of the sample agents. From analyzing the information, we also saw success especially against the agent called Boulware.

The main reason for that, as we see it, is that this agent uses a strategy similar to ours – compromise as late as possible. Because we have some advanced features, our agent still gets good results.

The success against Boulware is important for two reasons:

1. It seems to be the strongest sample agent.
2. We believe the other agents in the competition will behave more like the Boulware agent and less like other sample agents (for example, our agent acts more like the Boulware rather than any other agent).

In addition, we have seen that the opponent model is surprisingly good.

Although the frequency model sometimes inaccurate, it does predict the behavior of the opposing agent surprisingly well (see 2.2). Because we know the model might be inaccurate, our agent always gives the opponent bids some space/buffer from below so it does not skip an offer that might be good for the opponent. Therefore, our agent understands well what the opponent is looking for and knows how to adjust the offers accordingly.

We found that it is hard to isolate the affect of the learning rounds on the agent's performance, because the profiles of negotiations are different, and the diverse of the opponents vary during the tournament (for example, we might meet the hardliner 5 times in a row after a learning phase, but in some domains, agreement with the hardliner isn't profitable (its utility is much higher).

Nevertheless, we saw that in some domains an agreement cannot be achieved before learning phase, and we saw that the calibration of α helps the agent to achieve better utility against agents like the Boulware agent.

Last, we observed that the threshold (hyperparameters that determine the initial threshold – before the first learning phase, and the formula of ΔV) has huge effect on the agent's performance, so we tried some different values to get a better understanding of the effect. We didn't optimize the values because any optimization is only overfitting to the example agents and domains and practically meaningless towards the competition.

4. Conclusions

Throughout working on the agent, we learned some important points, but mainly exposed to the world of automated negotiation, include main techniques, architectures and models.

We cannot predict our agent's performance in the competition, but hope that it'll achieve good results. A lot of efforts was invested in this agent, hopefully making it a hard competitor. We think that some innovative and interesting ideas are inherent part of the agent (extended frequency model, opponent threshold model, adaptive point of compromising).

At this point – AlphaBIU is ready to compete. Wish it luck!

5. Code

AlphaBIU code can be found in this link.

6. References

[1] Baarslag, Tim, et al. "Decoupling negotiating agents to explore the space of negotiation strategies." *Novel insights in agent-based complex automated negotiation*. Springer, Tokyo, 2014. 61-83.

[2] Liu, Shan, et al. "An Automated Negotiating Agent that Searches the Bids around Nash Bargaining Solution to Obtain High Joint Utilities." *Annual Conference of the Japanese Society for Artificial Intelligence*. Springer, Cham, 2019.

[3] Baarslag, Tim, et al. "Predicting the performance of opponent models in automated negotiation." *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Vol. 2. IEEE, 2013.

[4] Zeng, Dajun, and Katia Sycara. "Bayesian learning in negotiation." *International Journal of Human-Computer Studies* 48.1 (1998): 125-141.

[5] Kraus, Sarit. "Automated negotiation and decision making in multiagent environments." *ECCAI Advanced course on artificial intelligence*. Springer, Berlin, Heidelberg, 2001.

[6] Mori, Akiyuki, and Takayuki Ito. "Atlas3: A negotiating agent based on expecting lower limit of concession function." Modern Approaches to Agent-based Complex Automated Negotiation. Springer, Cham, 2017. 169-173.