Automated Negotiation

Group MickVanGelderen

Mick Van Gelderen
mmvangelderen
4091566

January 22, 2015

Automated Negotation is the second assignment for the course Artificial Intelligence Techniques given at TU Delft. The assignment is to create a BOA agent using Genius, a tool to simulate negotiations between agents. This report covers the design and testing of this agent.

1 Going solo

Originally I was in group 7, however communication was lacking. Members were not responding to my questions even though they had seen them (thank you blue checkmarks). The deadline came so close that I had to make a choice: fail the course or do the assignment by myself. Seeing as I did the first assignment almost entirely by myself as well I wasn't going to let other people benefit from my work again.

Until today I had the impression that group 7 would be able to at least deliver something but as it turns out I was wrong. That, along with the pressure of deliverables for other courses, is why I did not announce this problem earlier.

2 High-level description of the agent and its structure

Usually, the key to writing a good agent is simplicity. That rule got me 2nd place for the GOAL agent so I will see if it holds for this assignment as well.

The Agent I am submitting is called Trig, because he likes trigonometry. Trig is a hard-headed agent with a strong decline towards its reservation value when approaching the deadline. Trig tries to exploit agents which do concessions even if he himself doesn't. To make it a bit harder to detect this fact, Trig generates random bids above his current acceptance level.

Trig is built with the following functions:

```
Bid generateBid() throws Exception
```

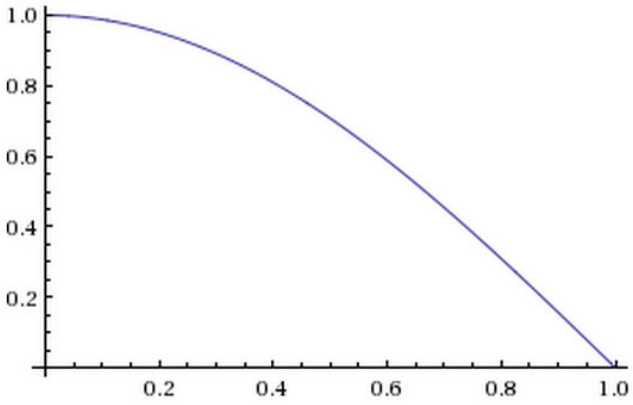

Figure 1: Cosine first quarter interpolator

This function tries to generate a random bid that is currently acceptable for the Agent. The acceptance level depends on the negotiation progress. If after a decent number of tries no suitable bid was found, it returns the maximum utility bid since any other bid might be unacceptable.

`boolean acceptable(Bid bid)`
This function takes the negotiation progress.

1. If the progress is less than $r\%$, it will only accept bids with an utility greater than the initial acceptance level. Currently I have settled on using $r = 80\%$.
2. Otherwise, it will accept bids with an utility greater than a value between the initial acceptance level and the reservation value. The interpolation function used for this is based on the cosine function.

`double cosDiscount(double progress, double high, double low)`
The interpolation is done as such:

$$\text{value} = \text{low} + (\text{high} - \text{low}) \cdot \cos\left(\frac{\pi}{2} \cdot \text{progress}\right)$$

The interpolation function can be seen in Figure 1.

3 Explanation of the negotiation strategy

The negotiation strategy that Trig applies is a weak Boulware stragegy combined with a random walker bidder. It tries to let offers float towards its preferences for as long as possible and when getting close to the deadline starts to concede more and more so an agreement can be made.

The advantages of this approach is that we do not care much about our opponents. As long as they don't walk away we do not have to act nice. By sending out randomized offers Trig might even cause simple opponents to not even realise that we are not being nice. In other words, we require no opponent model. I did try some experiments with the Bayesian Opponent Model (Koen Hendriks and Dmytro Tykhonov) but that experiment did not perform much better than Trig. Since I have only so much time choosing to further develop Trig was my best option.

The parameters for the Trig agent (that need to be adjusted in the code itself) are:

1. `declineStart = 0.8` The fraction of passed time after which the acceptance level will start to decline towards the reservation value.
2. `startUtility = 0.95` The initial acceptance level.

Further more it obviously depends on the preference profile which includes a reservation value parameter.

4 Tests

The tests were based on 3 agents: Boulware, Mick and Trig. The Boulware agent generates random bids until it finds one which it would accept, and accepts bids with utility $> .9$.

The Mick agent uses a Bayesian Opponent Model for its opponents. Its acceptance level declines while the negotiation progresses and it uses the opponent model to try and make a bid which:

1. the agent itself accepts and
2. is not much worse than the maximum of the expected utilities for the other agents for this bid.

In Figure 2 you can see what we would expect. Boulware gets the better end of this negotiation.

In Figure 3 we see that the nash product is higher because both parties did concessions.

Figure 4 shows that as is to be expected three Trig agents will come to an agreement around the 80% mark.

In Figure 5 it appears that the Mick van Boul agent do not reach an agreement causing the entire negotiation to fail.

To do proper tests it would be nice if the tournament functionality worked. Now I am getting a NullPointerException.

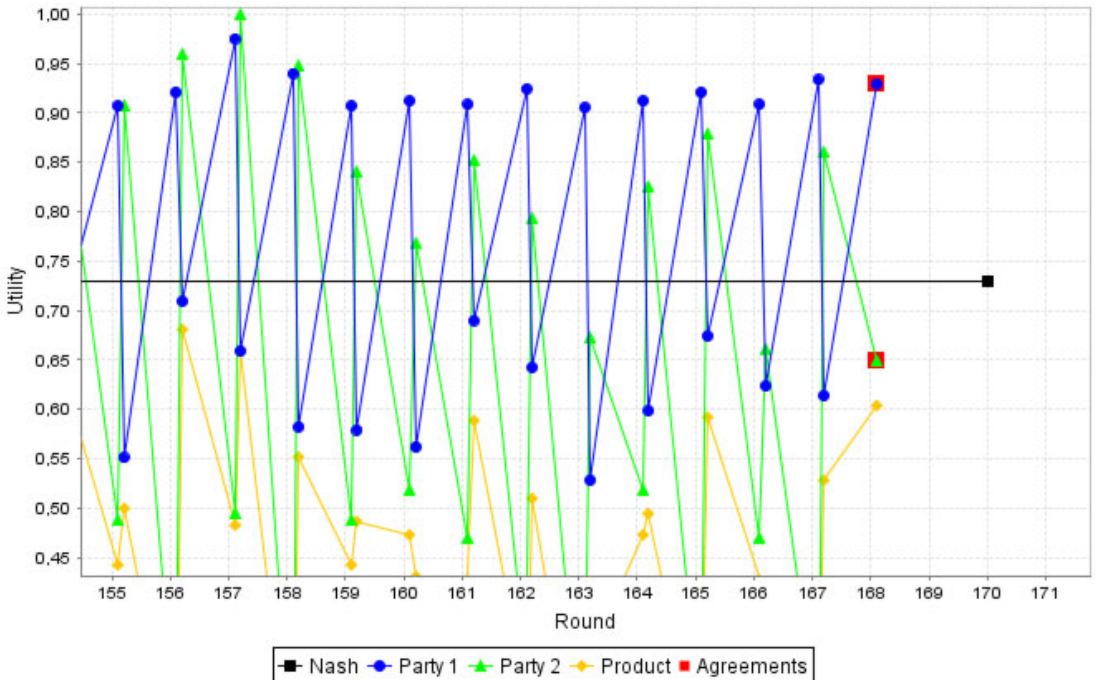

Figure 2: Boulware vs Trig

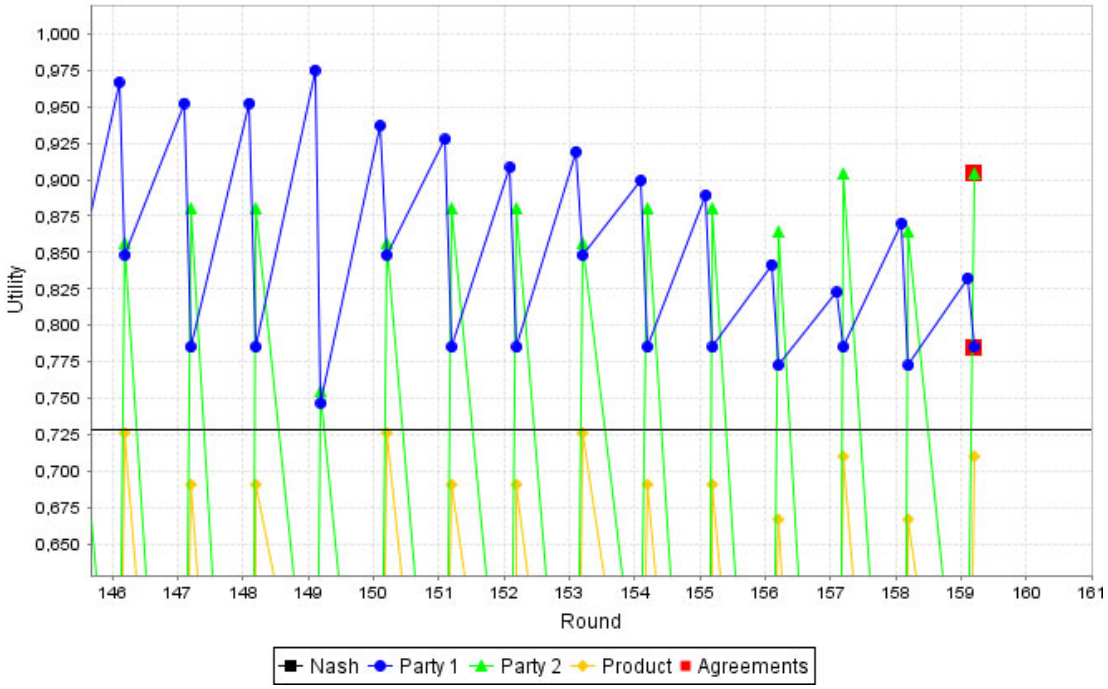

Figure 3: Trig vs Mick

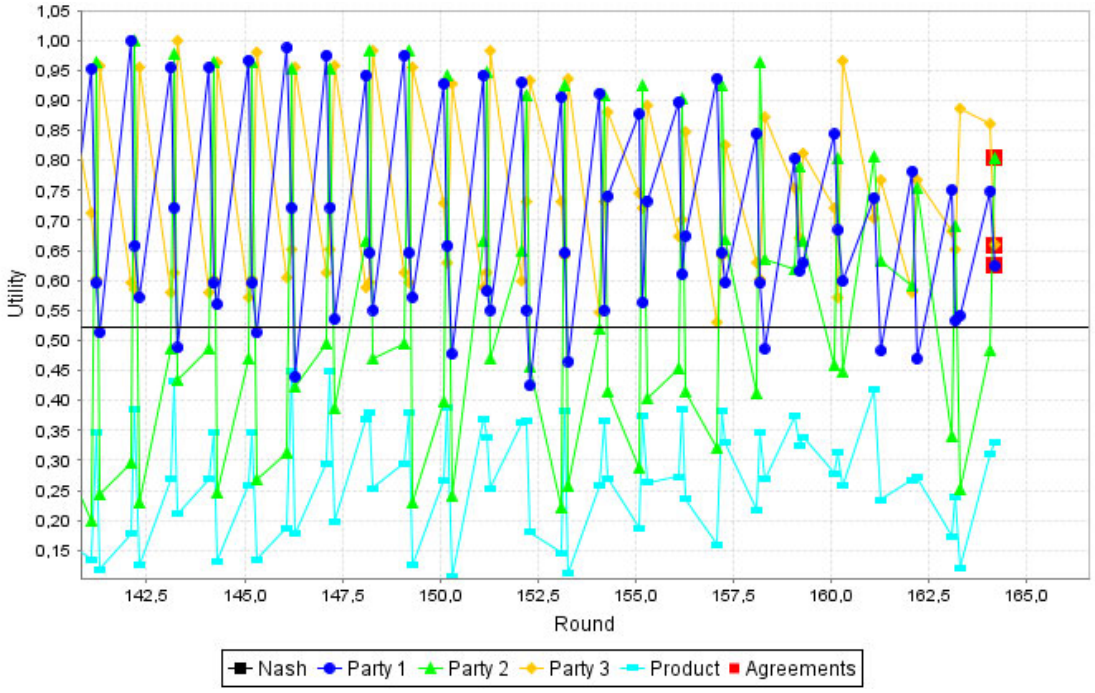

Figure 4: Trig vs Trig vs Trig

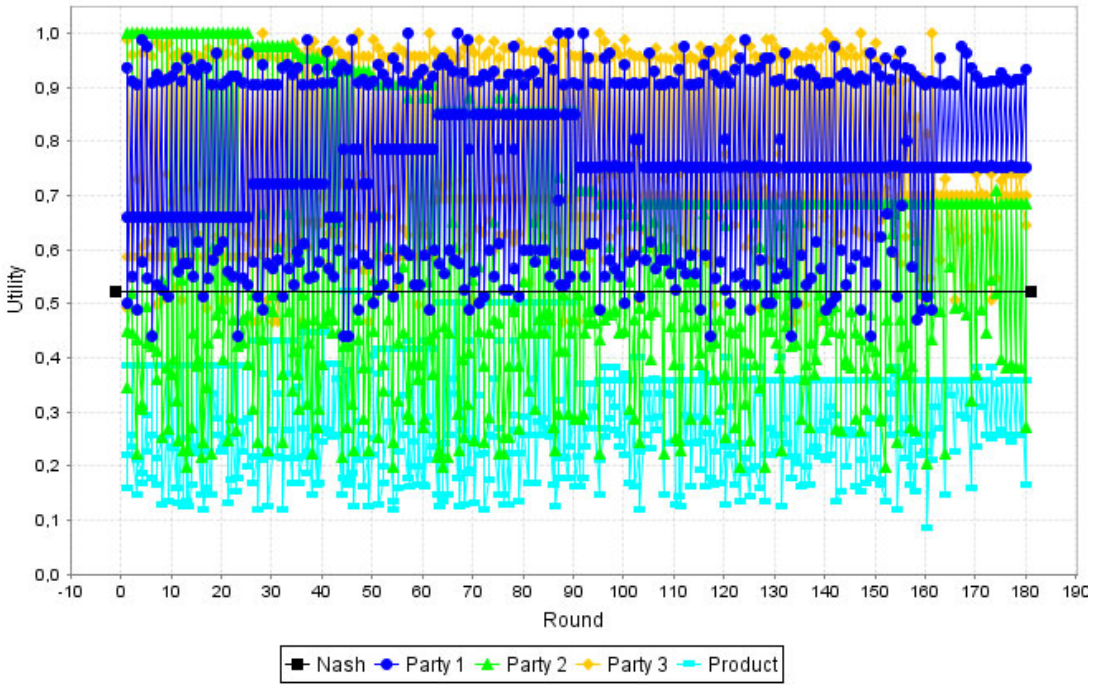

Figure 5: Boul vs Mick vs Trig

5 Conclusion

Depending on how the other students built their agents Trig might work very well, or if most of the others took nasty agents into account it will perform horribly.

Trig is too simple to replace humans. It does not really smartly abuse its BATNA/reservation value which, at least to me, the most important thing in negotiation bluffing how good your BATNA is and leveraging this perception. If your goal is to be liked, send an actual person. If your goal is to find a solution that evenly satisfies everyones wishes, interpolate the preference profiles/compute a bid that yields the most equal utility for each party.