Designing a Negotiating Party

Group 29

Joep Dumount - 4314271
j.c.dumont@student.tudelft.nl
TU Delft, Embedded Systems

Leon van der Knaap - 4972376
l.a.vanderknaap@student.tudelft.nl
TU Delft, Computer Science,
Data Science & Technology

Wouter Kok - 4169778
w.r.kok@student.tudelft.nl
TU Delft, Mechanical Engineering,
High-Tech Engineering

Robert Luijendijk - 4161467
r.luijendijk@student.tudelft.nl
TU Delft, Computer Science,
Data Science & Technology

Sven Uitendaal - 4478320
s.j.uitendaal@student.tudelft.nl
TU Delft, Mechanical Engineering,
Biomechanical Design

1 INTRODUCTION

In this report implementation of an agent to participate in automated negotiation sessions is discussed. The general purpose of a negotiation agent is to reduce the generally irrational actions of humans when negotiating, by providing pre-designed strategies, while aiming to optimize the obtained utility, based on the given (and known) preferences. For this reason, much research has been done in the field of automated negotiation. We refer to [3] for an extensive survey on the matter. The focus of our negotiation agent is that the strategies are dependent based on the turn of our agent. That is, either our agent is expected to start off the negotiation process by sending a bid (we call this turn 1); or our agent will be the first to receive a bid from the opposing agent (we call this turn 2). The decisions regarding accepting, bidding and opponent modeling may be dependent on the assigned turn.

First, in this introduction, we will provide a brief description of several components of a negotiation strategy and some existing implementations of negotiation agents. Next, we will discuss our implementation of the negotiation strategy, the bidding strategy and the opponent model. Note that these agent components were already introduced and included in previous assignments. We repeat them for completeness. Afterwards, we will present a strategy to deal with preference uncertainty: that is, how do we model issue weights and values if the actual values are uncertain? These components together make up our negotiation agent. We will then perform an extensive performance analyse and present obtained results based on a large number of negotiation sessions against varying agents, on multiple subject domains, with different utilities per party. Here, we will show that our agent indeed obtains different results based on its assigned turn. This analysis is followed by some general future perspectives on the topic of automated negotions, regarding the concept of mediators and possible improvements. We will conclude this report by discussing some possible directions for future improvements on our agent.

Negotiation Strategies

We will now briefly discuss some components of an agents negotiation strategies and some existing, basic examples of agents. These agents will be used later to compare the performance of our implementation.

Offering Strategy. An offering strategy is a mapping which maps a negotiation trace to a bid. The offering strategy can interact with the opponent model by consulting with it. [1]

Opponent Model. An opponent model is in the BOA framework a learning technique that constructs a model of the opponent's preference profile. [1]

Opponent Model Strategy. An opponent model strategy specifies how the opponent model is used to select a bid for the opponent and if the opponent model may be updated in a specific turn. [1]

Acceptance Strategy. The acceptance strategy determines whether the opponent's bid is acceptable and may even decide to prematurely end the negotiation. [1]

Existing agent strategies

- *Conceder (C)*: The conceder agent aims to make a concession (sending a bid that yields a lower utility for itself than its previously send bid) at every round. It does so in a determinstic pattern: if every possible bid were to be non-increasingly sorted based on its providing utility, the conceder agent will start with bid 1 in round 1, bid 2 in round 2, etc.
- *Hardliner (H)*: The hardliner agent does not make any concessions at all. It will send the bid yielding the highest utility at every turn. Futhermore, the hardliner agent will never accept a received bid. That means that the obtained utility will either be 1 (if its bid is accepted at some point), or 0 (when the negotiation is terminated at the deadline).
- *Boulware (B)*: The boulware agent makes use of an intermediate strategie compared to the two agents described above. In the early rounds, the boulware agent will not make concessions, acting as if it does not concede. However, once a sufficient amount of time has passed, the boulware will make some concessions.

2 IMPLEMENTED NEGOTIATION STRATEGY

The Acceptance Strategy

For the acceptance strategy we consider a combination of the following conditions.

AC_{next} : We start with accepting any bid of the opponent if it yields a higher utility than the bidding strategy. We take into account the opponents action combined with our own bid. This approach will yield a local optimum in the short term, by avoiding the possibility that the opposing agent will accept the direct counteroffer, yielding a smaller utility.

$AC_{combination}(T, \alpha)$: In this acceptance condition, two basic accepting conditions will be combined: AC_{time} and $AC_{const}(\alpha)$. The latter strategy accepts any bid yielding a utility larger or equal than α, whereas the first of these acceptance strategy traditionally splits the negotiation time into two phases: $[0, T)$ and $[T, 1]$, where, in the latter phase, the party is more prone to accept a bid yielding a relatively low utility. This strategy will result in a larger percentage of agreements.

We propose an adaptation on this strategy using a polynomial given variable T instead of a fixed value α, ie $AC_{time}[f(T)]$. We define function $f(T)$ as follows:

$$f(T) = (1 - D)\alpha(1 - T^{\beta}) + D \tag{1}$$

Here, α, β denote coefficients determining the shape of the function. Let D denote a dummy parameter which is equal to one if the agent which equals to one if our agent starts the first round, and zero otherwise. This allows for a more conceding acceptance strategy if our agent is not in a more dominant position. We will explain this concepts of turn-based tactics in detail in Section 2.

Therefore, given $D = 0, \forall\alpha, \beta,\ f(T)$ will yield α if $T = 0, f(T) = 0$ if $T = 1$ and $f(T)$ is decreasing over T. Hence, the agent that does not have turn 1, will accept any offer with a utility larger or equal to α, similar to $AC_{const}(\alpha)$ and yields every bid in the final round. Therefore, every negotiation will end in an agreement, in confirmation with AC_{time}. Furthermore, if $D = 1$, $f(T) = 1, T \in [0, 1]$. This implies that the agent who starts the negotiation has an acceptance strategy equivalent to AC_{next}.

Currently, we use parameters $\alpha = 1, \beta = 25$.
We combine the acceptance conditions as
$AC_{next} \cup AC_{combination}(T, \alpha)$. That is, we accept the received bid if at least one of the conditions hold. In addition to the turn of the agent and the opponent's most recent offer, this strategy also considers the (time until) the deadline.

The Bidding Strategy

We will adapt our bidding strategy, from the general random bidding strategies. Given a certain threshold, we want to send a random bid with a utility larger or equal than this threshold. Instead of generating completely random bids and only sending them if they meet the threshold, we will first establish all possible bids and then choose one at random. If no possible bids above our given threshold exists, we will iteratively lower the threshold by a fixed values, until one or more bids exist. This adaptation guarantees that every bid will satisfy the threshold. Hence, it is not possible to send an undesirable bid if no sufficient bid is found over time.

Similar to the acceptance strategy, the values of the thresholds are directly dependent on whether or not the agent has the starting turn every roudn (see Section 2) . In the bidding strategy, however, they remain constant over time (ie 0.7 and 0.9 for agent having turn 1, turn 2, respectively).

Turn based negotiations

In this section we will we make some observations regarding the turn-based protocols of this assignment. We will then use these observations to support our previously derived strategies.

The negotiation is played in rounds, given this fact we could let the strategy depend on the order of turns per party. The stated protocol dictates that the party that starts off the negotiation (in the first round), will start each next round. For this negotiation we could lend insights from the field of game theory for this sequential, dynamic game. In accordance with the previous sections, let $D = 1$ denote the agent that has the first turn in every round, 0 otherwise.

Hence, we have a dynamic game with a known deadline and a pay-off of zero for both agents if no agreement is reached at the deadline. Game theory suggests that therefore, if the turn of the second agent in the last round is reached, the last agent effectively gets to choose between the utility attained at the last received bid and a utility of zero. In this case, the utility of zero would imply no agreement is reached. Since an agent aims to optimize its personal pay-off, this situation will pressure any party to accept the last offer. Retroactively the agent, having $D = 1$, can utilize a bid with an higher utility in the last round than previous bids would suggest, given that the opposing agent is more willing to accept.

According to the concept of the subgame perfect nash equilibria (SPNE) (see eg [4]), we can apply dynamic backtracking for each subgame in a dynamic game to find the 'optimal' choice. This will result in the observation that the agent for which $D = 1$ can generate bids with a larger utility and accept less bids from the opposing parties, and still end up in an agreement.

This is visualized in Figure 1. Consider two parties who both comply to our described strategies. If Agent 2 gets a turn in the last round of the negotiation, it will always accept, since his own pay-off of 'accepting' (v_2), is strictly larger than the pay-off of terminating (0).

Given this information, if Agent 1 gets a turn in the last round of the negotiation, he can use this information when choosing the action that yields the highest pay-off. This is either 'accepting' yielding v_1, or 'bidding', which gives $\hat{v}_1$. Since agent 1 can observe its own utilities for both bids, the two can be compared and therefore the action yielding the maximum can be chosen with certainty. This is embedded in the acceptance strategy AC_{next}. The turn prior (turn 2 in second-to-last round), Agent 2 can either accept, yielding utility equal to v_2, or generate a new bid. This second option is the first option we encounter when backtracking yielding any uncertainty. The obtained utility for Agent 2, in case he does not accept, is dependent on the remaining action of Agent 1. We can generalize this notion of uncertainty back to the beginning of the

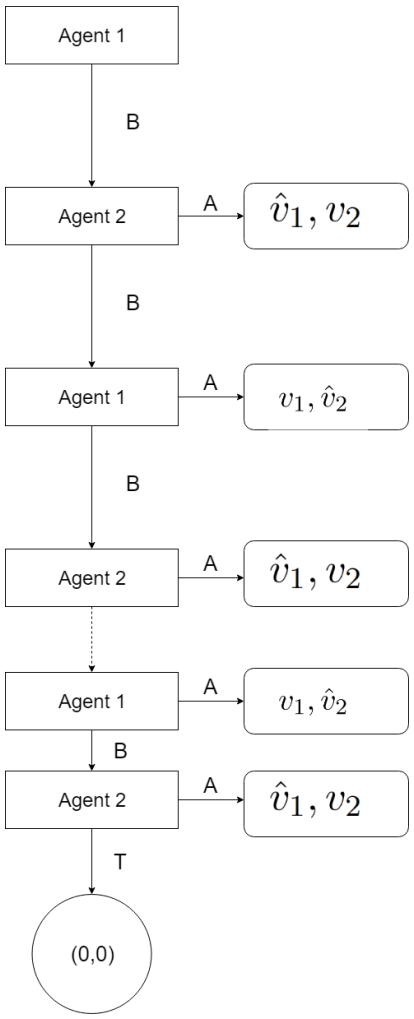

Figure 1: Flowchart of the obtained utilities based on the actions of both agents. B denotes the action of a sent offer; A represents the accepting action of an agent; T denotes a terminating action (no agreement). The obtained utilities for both agents are given in the terminating nodes, where v_i denotes an observed utility for agent i, and $\hat{v}_i$ denotes an unobserved utility.

negotiation. Therefore, we can now conclude that the agent who has the first turn of the negotiation, will be in a more dominant position throughout the entire negotiation. We use this notion in both our acceptance and bidding strategy using dummy parameter D.

For an extensive survey on dynamic game theory, we refer to [2].

HardHeaded Frequency Model

The issue with the current *HardHeaded Frequency Model* is that it does not take the different rounds of the negotiation into account. That is, the estimated issue weights and issue values are updated with a coefficient that is constant over time. Many agents will use a bidding strategy starting with bids that yield a large utility for itself. Over time, if an agreement is not yet reached, concessions will be made resulting in bids that generate lower utilities. Hence, observing differences between received bids early in the negotiations, reveal more information about the underlying issue weights and values of the opponent.

Issue weights When estimating the issue weights, the *HardHeaded Frequency Model* increases the estimated issue weights of an issue in case the issue values remain unchanged using a parameter n. This parameter is constant over time, implying that a changed value in the last round of a negotiation is just as determinative as changing a value from turn one over to turn two.

Issue values When estimating the issue values, the *HardHeaded Frequency Model* increases the (unnormalised) estimated issue values if the issue value is present in the bid by incrementing by a given parameter. This parameter (usually equal to 1) is fixed and therefore also constant over time.

Time-dependent HardHeaded frequency model We propose to adapt the *Hardheaded Frequency Model* by adding a time parameter $T \in [0, 1]$, with $T = 0$ and $T = 1$ denoting the first round and the deadline of the negotiation, respectively. This parameter is used in estimating both the issue weights and the issue values as follows:

Issue weights We adapt the formula of the standard *HardHeaded frequency model* Equation 2 to 3. Here $(1 - T)^\gamma$ is a correction term over time, yielding the original formula if $T = 0$, and yielding $W_i^{new} = W_i^{old}$ when T approaches 1, for all values of γ. The correction term is polynomially decreasing over γ ($\gamma = 1$ yields a linear slope). This implies that we assign significantly more value to bids near the start of the negotiation. *We currently use* $\gamma = 2$.

$$W_i^{new} = W_i^{old} + n \; for \; issue \; i \tag{2}$$

$$W_i^{new} = W_i^{old} + n \times (1 - T)^\gamma \tag{3}$$

Issue values Instead of counting the number of appearances for every issue values throughout the negotiation and normalising this, we once again want to assign larger values to frequently occurring issue values at the start of the negotiation. We evaluate the appearing issue values per receiving bid by $\lceil (D(1 - T)^\zeta \rceil$, with D denoting the deadline of the negotiation, in terms of rounds (that is, the round for which $T = 1$). Hence, in the first round of the negotiation, implying $T = 1$, every appearing issue value is counted D times, before being normalized. *We currently use* $\zeta = 2$.

Opponent model strategy

So far, in our bidding strategy, we have not considered the estimated utility of our opponent. We had generated a list of all bids yielding a personal utility above a given threshold. Every round, one bid will be offered at random. We aim to improve the proportion of accepted bids by using the described opponent model, while still only sending bids that result in a high own utility.

Now, instead of choosing a random bid from the list, we rank the list based on the *estimated Nash products*. We define the estimated Nash products of bid i at time T as $\pi_{iT} = u_i^A u_{iT}^B$. With B as the opponent, A as ourselves and u as the utility. Note that u_i^B is known and constant over time. Since u_{iT}^B, denoting the utility of bid i at time T of the opponent, is an estimated value, π_{iT} is an estimation as well. However, it still holds that if $\pi_{iT} > \pi_{jT}$ then $u_i^A > u_j^A$ or $u_{iT}^B > u_{jT}^B$ for bids i, j. This means that if we consider to send a bid i yielding a smaller personal utility than bid j, we have estimated that bid i is preferred by our opponent.

We use the estimated Nash products of all bids that exceed our personal utility threshold as follows:
Instead of constantly selecting the first bid of the list, we send a bid based on time, our own utility and the Nash product. Because we base the bidding on time, the initial bids that we send have a high utility for us. Over time, when we get more information about our opponent we have a higher chance of sending him a Nash product bid. This happens with the formula: $l = (1 - time) * len(totalbids)$. This l(=length of the sublist) is then used to pull a random bid from the sublist of all the sorted bids. If $l = 0$ it will pick the highest sorted bid. This way, when time passes, there is a higher chance to pull a better bid for the opponent, so on average we concede to the opponent. A more formal formulation summarizes the strategy in equation (4) , with bid index r from the sorted Nash product list. And the length of the bids above the target (which is an set threshold), $|S|$.

$$r \sim U[0, (1 - T) \times |S|] \tag{4}$$

2.G: Preference uncertainty

The estimation of the uncertain utility space is done in a similar way as the estimation of the opponent model. However instead of using the opponents bids, a list of bids is obtained from the `userModel` variable where the order of preference is known to be increasing. At first, the order of the bids is reversed such that the first bid is the bid with the highest preference and the preference of bids goes downwards.
For the estimation of the uncertain utility space each issue needs a weight and each value of each issue (here named issuevalue) needs a utility. To estimate those there is iterated through the list of bids and scores are updated for the weights and issuevalues. The scores for the weights are initialized at 1/(the amount of issues) and the scores for the issuevalues at 0.
For each bid the weights are updated by comparing the value of each issue in a bid with the value of that issue in the previous bid in the list. If the value has remained the same, the weight for this issue increases by a value `weightterm`. This increase in weights is done because it is likely important, as it wasn't changed in the bid. The issuevalues scores are updated when a value appears in a bid. The variable `valueterm` is added to the score of that value. This is to keep track of how often a value occurs in the bids.
Both variables `weightterm` and `valueterm` can be set as a constant value for every bid that is iterated through. However since the first bids are known to be more preferred, their issues should weigh harder in determining the utility. Therefore the absolute value that is added to the weights and the scores of the issuevalues should decrease while iterating through the ordered bids. These absolute values are calculated with the formula

$$\text{Increase} = ((s - i)/s)^{\zeta} \tag{5}$$

where i is the iteration, s = length of bids list + 1 and ζ is a correction term over time. For i between 0 and s the increase value is between 0 and 1. To determine which ζ should be used for

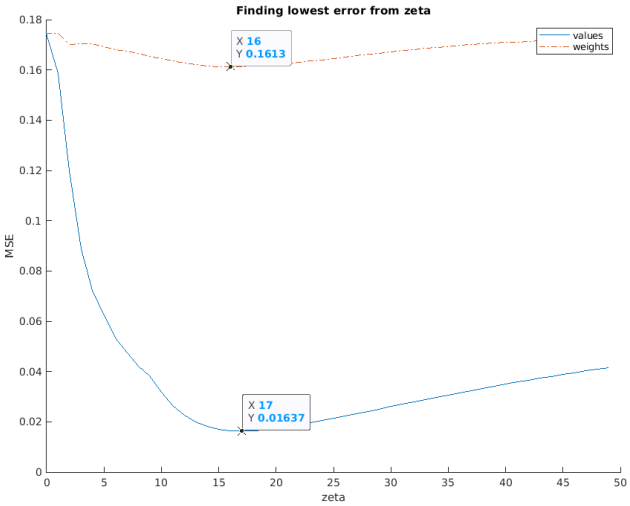

Figure 2: The obtained MSE from changing the zeta from equation 5 with the lowest MSE displayed. First iteration where the other variable is assigned ζ = 0

`weightterm` and `valueterm` an iterative script is written to measure the error resulting from setting ζ to a certain value. The error is measured as the mean squared error (MSE) between the estimated utility and the real utility (which can be obtained for experimental purposes) for all the bids that are given from the bidranking. The goal is to get the lowest error so that the estimated is the closest to the real utility function. When testing for the lowests ζ, the other variables' ζ is kept at 0 (resulting in a constant 1 from eq 5). For the tests the party domain 2 with 50 rankings are used and result in the errors that can be observed in Figure 2.
The MSE can be seen to change a lot by tuning the ζ. For the `valueterm` the lowest value is found at $\zeta = 17$ and for `weightterm` at $\zeta = 16$. With these settings the MSE goes as low as 0.01637 where it started at 0.1745, when a constant value was used ($\zeta = 0$).
An even lower MSE can be reached if there is iterated over the ζ for one variable when the other is now set at the value found from Figure 2. These results can then again be used to obtain a lower value until the lowest MSE doesn't change anymore. The final iteration can be observed in Figure 3, which took 3 iterations to get to. For the `valueterm` the lowest value is found at $\zeta = 14$ and for `weightterm` at $\zeta = 4$. Resulting in an MSE of 0.007094 making the use of equation 5 almost 25x as accurate as using a constant value.
The relatively high ζ (higher than 1) means that using only the

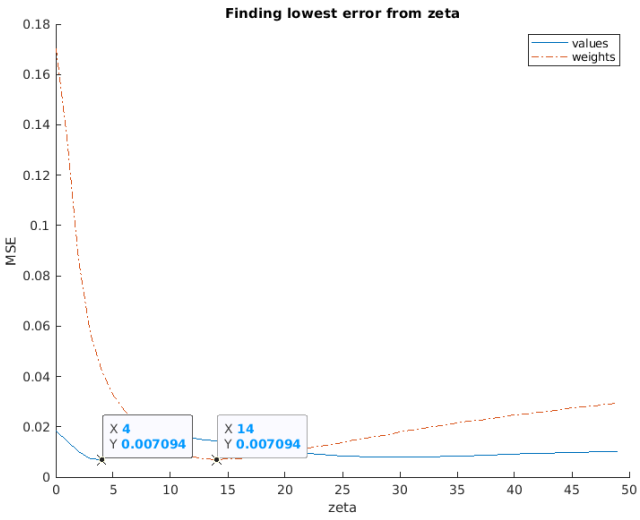

Figure 3: The obtained MSE from changing the zeta from equation 5 with the lowest MSE displayed. Final iteration where changing the ζ in either variable does not result in a better MSE.

first few bids are used to determine the preference. When less bids are given the MSE increases, which is to be expected since the estimation can be performed on less bids. For a ranking of size 10 the MSE is 0.043979 and for a size 20 the MSE is 0.022430 within the same party domain. For the other party domain, 1, the MSEs are 0.049840 for size 10, 0.032876 for size 20 and 0.005503 for size 50, making the approximation very good since the errors are low.

Implementation of the source code

In this section we will briefly describe the (methods in) the source code that make up our implementation of the negotiation agent.

2.0.1 AC_ai29. In this class, the acceptance strategy is implemented. With an incoming bid, the function *determineAcceptibility* is called, which determines if the bid should be accepted or not, following the Acceptance Strategy. Returns an Action *Accept* or *Reject*.

2.0.2 BiddingStrategy29. In this class, a new bid is produced. When the negotiation asks for a new bid, the function *determineNextBid* is called. The returned bid is produced following the Bidding Strategy.

2.0.3 OpponentModelAI29. In this class, the bidding strategy of the opponent model is analyzed to estimate the opponent's utility space. When a bid comes in. it is passed to the function *updateModel*, together with the time. The model is then updated according to the way described in the HardHeaded Frequency Model.

2.0.4 OpponentStrategyModel29. Based on the opponent model, a bid is created. In this function, a list is passed with all possible bids. A bid from this list is chosen using the method described in Opponent model strategy. This bid is returned.

2.0.5 Group29_BoaPartyPreferencedUtility. Based on a list of bids sorted on preference, this class has a function which estimates the own utilityspace. This is done in the way described in 2.G: Preference uncertainty.

2.0.6 BidDetailsNash. This class is used to compare bids in terms of Nash products. This is done in the function *compareTo*. Another BidDetailsNash is passed. Returns 1 if the Nash product is smaller, returns -1 if the Nash product is larger and returns 0 otherwise.

3 PERFORMANCE ANALYSIS

In this section we will discuss an extensive analyse on the performance of our negotiated agent, based on negotiation sessions against a set of varying agents. This set consists of the agents *Boulware, Conceder* and *Hardliner*, all available in the Genius framework. The basic strategies of these agents are described in the introduction of this report. In addition, we analyse the performance against our own agent. That is: a negotiation between two instances of our own agent. Since the strategy of our agent is dependent on its (starting) turn, we will present results for the cases of turn 1 and turn 2. We will first discuss the performance obtained on the *party* domain, followed by the *Jobs* and *Laptop* domains. Within each domain, we have considered all available utility profiles (per agent) available on Genius.

We will first analyze the obtained results based on the utilities of the negotiation outcomes within the party domains. In Figure 4 we show the obtained utilities of our own agents (referred to as *AI29* or *'us'*) against the aforementioned agents, taking both starting turns. As can be seen in Figure 4, if we are player 1, our strategy makes sure our obtained utility is always larger than 0.9, which is the pre-set threshold. This is because we only offer bids which have a utility of 0.9 or higher and because in the final round we get the last opportunity to make a bid, the opponent only has the choice to accept our bid or walk away. This works against every party which does not want to walk away without an agreement. That is why this strategy doesn't work against opponents like the Hardliner, which sticks to it's best possible bid and never accepts a different bid.

If we are player 2 we only offer bids which are larger than 0.7, because we know we are in the disadvantaged position. Often an agreement with utility ≥ 0.7 takes place, because apparently there is often a bid with that utility for us, with an high enough utility for the opponent. Sometimes we get a utility of < 0.7, because when time passes, we are more eager to accept bids. This is especially valid against the Hardliner. Because he sticks to it's best bid we will accept it when time passes.

In Table 1 we summarize the average obtained utilities for both agents between any pair of the four agents.

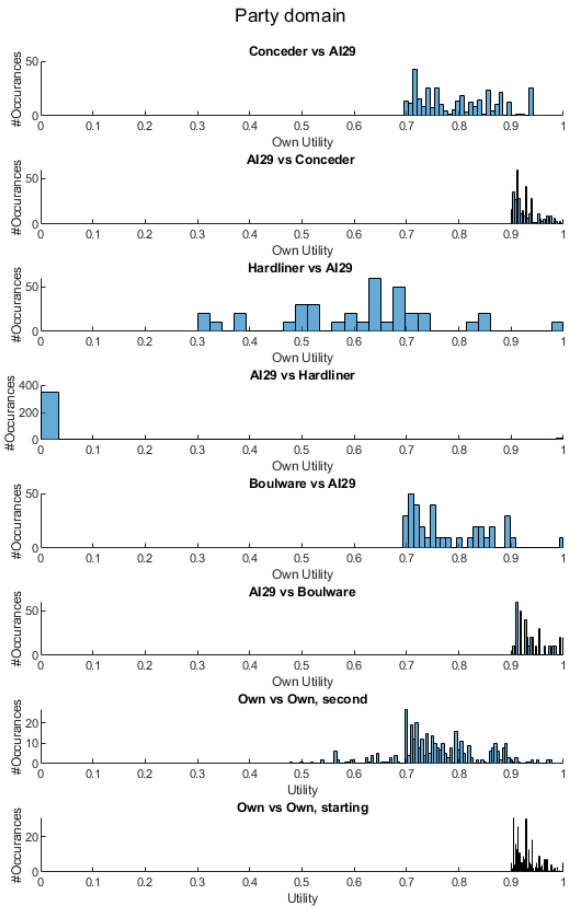

Figure 4: Obtained utilities of the negotiation outcomes between our party vs different party within the party domain.

Table 1: Average obtained utilities for both agents (U_a, U_b) for any pair of agents. The agent in turn 1 is stated in the rows; the agent in turn 2 is stated in the columns.

	B	C	H	us
B	(0.80, 0.81)	(0.99, 0.64)	(0.61, 1)	(0.94, 0.78)
C	(0.64, 1)	(0.80, 0.81)	(0.61, 1)	(0.82, 0.80)
H	(1, 0.61)	(1, 0.61)	(0.03, 0.03)	(1, 0.61)
AI29	(0.94, 0.8)	(0.93,0.77)	(0.03, 0.03)	(0.93, 0.76)

From Table 1 we can make some similar observations as based on Figure 4. That is, our agents performs especially well when he has turn 1, except for the case against the hardliner agent. However, we can observe that the Boulware and Conceder agents are able to obtain larger utilities against the Hardliner. This is due to the acceptance strategies of both agents, stating that they will accept at some point, which is somewhat similar to our strategy in turn 2. Since the only bid that can be accepted is the initial bid of the hardliner, the obtained utilities for both agents are equal in several combinations (i.e. (0.61, 1) or (1, 0.61). The bad performance in turn 1 against the hardliner agent is due to a conflict with our assumption about the acceptance strategy of agents in general. We have argued that it is never a good strategy to not accept a bid when there is no option to send a counter bid anymore. Although the hardliner agent seems to perform well against our agent, its average obtained utilities against all agents is actually the lowest. This can be seen in the upper part of Table 2. This table contains, for every agent, the average obtained utilities against all opposing agents. From this table, we can conclude that the hardliner agents performs especially bad in turn 2. This is due to 50% of its opponents not accepting its only bid.

Table 2: Obtained utilities per agent for both turn 1 and 2, averaged over the different opposing agents. The upper part includes the average utilities between all agents, the lower part excludes the hardliner agent.

All agents	turn 1	turn 2
B	0.835	0.805
C	0.7175	0.7075
H	0.7575	0.515
AI29	0.7075	0.7375
Excluding H		
B	0.91	0.87
C	0.75	0.74
AI29	0.93	0.78

Since the acceptance strategy of the hardliner agent contradicts our previous assumption and we have shown that it does not yield desirable results, we have also provided the average utilities excluding the hardline agent in the lower part of Table 2. Here we can observe that our agent obtains the best average utilities in turn 1 only. In turn 2, however, the obtained utilities for our agent are relatively small. These obtained utilities are mainly determined by the initial acceptance threshold of 0.7. In Figure 5, we show the number of rounds of each negotiation against the different agents, in case our agent has turn 2. From this figure we can deduce some information about the different negotiations:

- Against hardliner: we either accept opponents bid immediately or wait until very shortly before the deadline to accept
- Against conceder: we either accept opponents bid immediately or quickly thereafter since the opponent concedes quickly (or opponent accepts our bid)
- Against boulware: we either accept opponents bid immediately or pretty late since the opponent concedes very slowly (or opponent accepts our bid)
- Against AI29: we either accept opponents bid immediately or pretty evenly throughout the negotiation.

Against all agents, we seem to accept the first received bid in several negotiations. This result might be diminished by increasing the acceptance threshold. However, doing so might lead to a significantly

worse performance on a completely different domain, consisting of more issues.

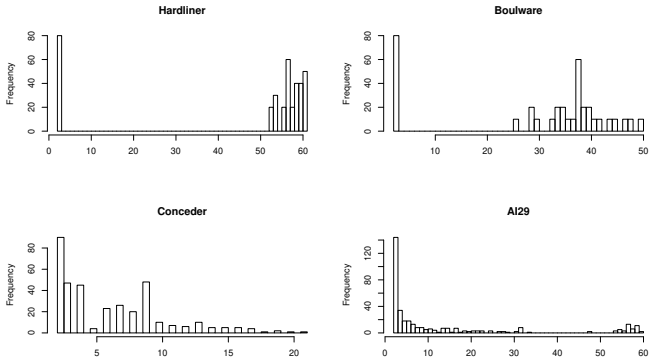

Figure 5: Number of rounds of each negotiation session against between varying agents (turn 1) and our agent (turn 2).

3.A: Party domain

3.0.1 Performance explanation. As can be seen in Figure 4, our party behaves kind of the same when negotiating against a Boulware or a Conceder party. When we start the negotiation our outcome utility is always ≥ 0.9, because we are hardheaded in this threshold. When we start second we try not to go lower than 0.7 in first place. We try to make a quick deal which is good enough. If we find out this deal is not accepted by the opponent, we will slowly concede by lowering the bidding threshold and lowering the acceptance utility. On average, this gives us a lower utility when we start as the second player than if we start as the first player. However, because we are slowly checking what is the maximum outcome for us it still gives us a better result than the Conceder party as can be seen in Table 1. Apparently there is always a bid which has a utility of ≥ 0.7 for us and a high enough utility for the opponent for an agreement.

When we play against the Hardliner party and we are the starting party, both us and the opponent are too *stubborn* to concede, so unless the bid has the best possible utility for the opponent and a utility of ≥ 0.9 for us, there comes no agreement. If we start second we are the party which concedes, so in the end we will accept the bid the opponent offers, no matter what the utility is for us.

If we play against our own party if we start we always reach a utility of ≥ 0.9, like described earlier. When starting second we try to get an agreement with a utility ≥ 0.7. In Figure 4 it is easily seen that a lot of utilities are larger than 0.7. If we find out that this is not possible, we will accept bids with a lower utility for us.

3.0.2 Nash product. According to Table 3 an outcome on the Nash product is possible. It happens a lot when negotiating against the Boulware opponent. This is because both parties hardly like to concede, so in the end they come out in a point which is not optimal for both, but lies on the Pareto frontier: the Nash product.

The Conceder opponent like to concede a lot. Because we like it less to concede it is more likely to reach an outcome which is better for us than for the Conceder opponent and therefore less likely to reach the Nash product.

The Hardliner opponent stick to it's best bid. Therefore an outcome on the Nash product takes only place if the Nash product is the best bid for the opponent.

When playing against the same party as us the starting player sticks to bids with utility ≥ 0.9. Therefore it is only possible to reach an outcome on the Nash product if this point is a bid for which one utility is at least 0.9.

Table 3: The percentage of negotiation sessions within the *party* domain in which the obtained outcomes are the nash product, on the pareto frontier, and end up in an agreement.

turn 1	%nash	%pareto	%agreement
B	41.67	80.56	100.00
C	10.56	39.72	100.00
H	2.78	2.78	2.78
Us	6.94	37.78	100.00
Tot	15.49	40.21	75.69
turn 2			
B	16.67	88.89	100.00
C	0.28	15.56	100.00
H	2.78	100.00	100.00
Us	6.94	37.78	100.00
Tot	6.67	60.56	100.00

3.0.3 Efficient outcomes. When looking at efficient outcomes there is looked how many outcomes are on the Pareto frontier. This can be explained by the following. When playing against the Boulware opponent this happens a lot. Because both us and the opponent are hardly conceding, often an outcome which is good for both takes place. If an outcome is good for both there is a good chance this outcome lies on the Pareto frontier.

Against the Conceder party there are also outcomes on the Pareto frontier, but less often than against the Boulware party. This can be explained by the fact that the Conceder party concedes fast, which leads to a suboptimal outcome for this party. If we start as the second player, we are also more likely to concede, so there will be diverged more from the Pareto frontier.

If we start against the Hardliner party we only get a deal if the bid has the best utility for the Hardliner party and a utility of ≥ 0.9 for us. This happens to be always on the Pareto frontier, so if there comes an agreement it will be on the Pareto frontier. If we start second we will always get a deal. Because this deal has the highest utility for the Hardliner opponent it is on the Pareto frontier.

Against our own party we always get a deal which has a utility of ≥ 0.9 for the first player. The second player wants an agreement of at least 0.7 utility for him, so it happens sometimes that the agreeing bid lies on the Pareto frontier, however, there can also be bids which satisfies this condition for both parties which don't lie on the Pareto frontier.

In the appendix all of the graphs, Figures 14-22, are visible of the sessions that took place in the party domain. In the legend of these figures, the party that is first in the legend is the party that starts the negotiations and the other party is second in the negotiations. For example in Figure 14, Boulware goes first and our agents goes second. In this section it is clearly seen that the Hardline model, Figures 18 and 19, holds on to his one utility, because it stays in one place. It is also clearly seen that there is no agreement reached when our agent starts. This is because we have a threshold on 0.9, this is also clearly seen in all of the figures. From the Figures 16 and 17 it shows clearly that we will always get an agreement because it Concedes always. The same is to be seen in Figures 14 and 15, where the Boulware model will concede near the end of the negotiations. When looking at Figures 20, 21 and 22 it will also show that when our agent is playing against itself it will always come to an agreement, because of the conceding nature of our agent when it goes second.

3.B: Laptop and jobs domain

In this section we discuss results obtained by our negotiation agent on a different domain. So far, all negotiations were run within the party' domain. For the remainder of the section, we will look at the 'jobs' and the 'laptops' domain. In Figures 6 and 7 we present the obtained utilities for our agent against all previously described agents, for both turns. Note that these Figures are similar to Figure 4. However, the jobs and party domains consist of less utility profiles within the Genius framework.

The performance in the laptop and jobs domains is quite similar to what is found for the "party" domain, making it perform somewhat generic on other domains. Opponent negotiating agents with the intent to create an agreement get a deal with a high utility for our agent. This is true when we are the starting negotiator or when the utility space allows this with us as second player.

The hardliner does not concede and therefore does end up with a lot of outcomes where no deal is reached, which explains the peak in Hardliner vs AI29 in the jobs domain. When we start the negotiation in turn 1, we will not accept any bid with a utility for us lower than 0.9. Therefore it is difficult to create a deal with the Hardliner party in this setting. As the second player we are conceding, making our utility very utility space-dependent.

In the end our performance depends on 3 things, dependent on if we are the starting player or not. When we are the starting player we have a utility of ≥ 0.9 if the opponent doesn't walk away in the end, meaning the opponent is always prepared to make an agreement. When we are the second player our performance depends on the speed of conceding of the opponent and the utility space. A domain which has more possible outcomes, meaning the domain consist of more issues, gives a less binary outcomes. When we compare the party domain which has many issues, to the jobs domain we can clearly see the difference in distribution and performance for player 2.

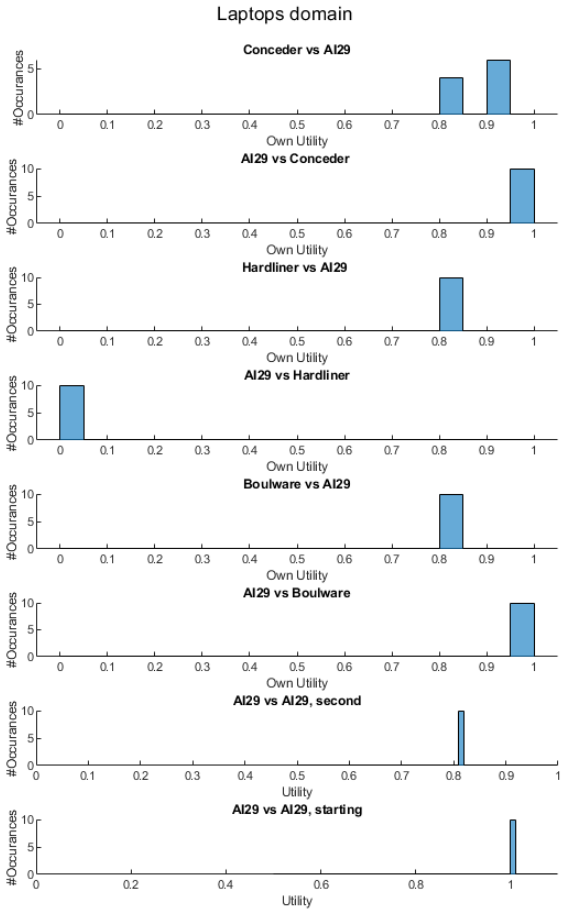

Figure 6: Histogram of the session of a tournament sorted on Agents competing.

A lower distance to the Pareto frontier is desirable, looking at Figures 13, 9 and 11 the agent is able to achieve low values. The Nash product (see Tables 4, 5) indicates a positive outcome for both parties, the closeness to this point indicates how well the negotiation process went for both parties. The party domain, which has a lot of issues, is not able to achieve a low distance, having too much spread. While the other domains yield a acceptable low value against the different agents.

FUTURE PERSPECTIVES

In this section we will briefly discuss the concepts of mediators and possible improvements to automated negotiation.

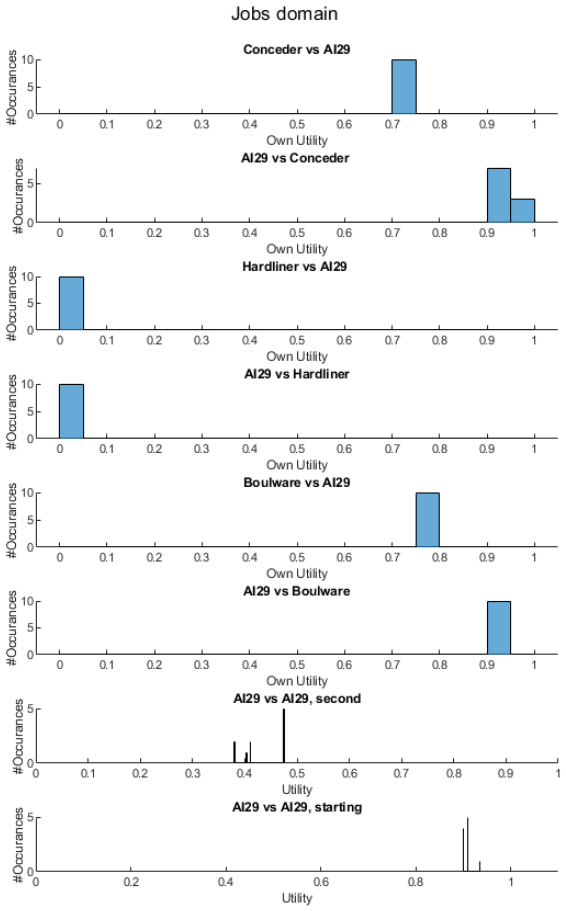

Figure 7: Histogram of the sessions of a tournament sorted on Agents competing.

4: Mediators

The idea of a mediator is that there is a trusted third party involved. There are different versions of such a mediator, but the idea is the same. Both parties will communicate through the mediator. The parties will send their bids to the mediator, the mediator will collect their bids and will send an answer back to the parties. This is where the mediators will differ, and we will discuss two common ones.

(1) The simple mediator protocol. This mediator protocol will send an offer back to the parties, where the parties will either accept or reject his offer independently. If one of the parties rejects, the mediator will send a new random bid, this will go on until all of the parties accept the same offer, or if the deadline is reached.
(2) The mediator feedback protocol. This protocol will take feedback into account of the parties when a bid is send out by the mediator. The parties will send their feedback back to the mediator about the comparison between the last and the previous to last bid. The bid is accepted if the bid did not receive negative feedback.

Table 4: The percentage of negotiation sessions within the *jobs* domain in which the obtained outcomes are the nash product, on the pareto frontier, and end up in an agreement.

turn 1	%nash	%pareto	%agreement
B	0	100	100
C	0	90	100
H	0	0	0
us	0	60	100
tot	0	62.5	75
turn 2			
B	0	0	100
C	0	0	100
H	0	0	100
us	0	60	100
tot	0	15	100

Table 5: The percentage of negotiation sessions within the *laptop* domain in which the obtained outcomes are the nash product, on the pareto frontier, and end up in an agreement.

turn 1	%nash	%pareto	%agreement
B	100	100	100
C	100	100	100
H	0	0	100
us	100	100	100
tot	75	75	75
turn 2			
B	100	100	100
C	40	100	100
H	100	100	100
us	100	100	100
tot	85	100	100

These mediators will not help to get an acceptance faster, because the simple mediator protocol will only send random bids. This will eventually result in an acceptance, but this is not reliable and will do nothing to get the parties closer on the subject. The feedback protocol will have some merit, but if parties will have a certain threshold where they will not dive under or above, it will become useless and will go on till the deadline. This is different when both parties already have their bids in the same area. In that case the mediator feedback protocol can guide both parties to an acceptance bid, this will be done in less time than when the parties will negotiate between them.

Another way a mediator could be helpful is when both parties already have send their highest value in a domain. The mediator can send to each of the parties that this is the case. It would not be

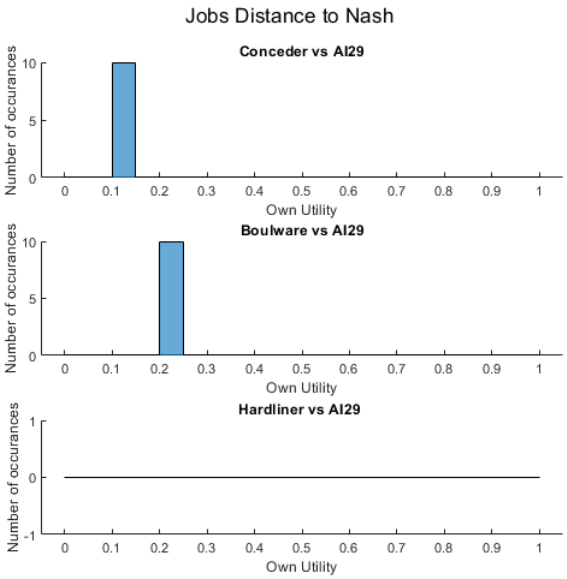

Figure 8: Histogram of the Nash product sorted on Agents competing.

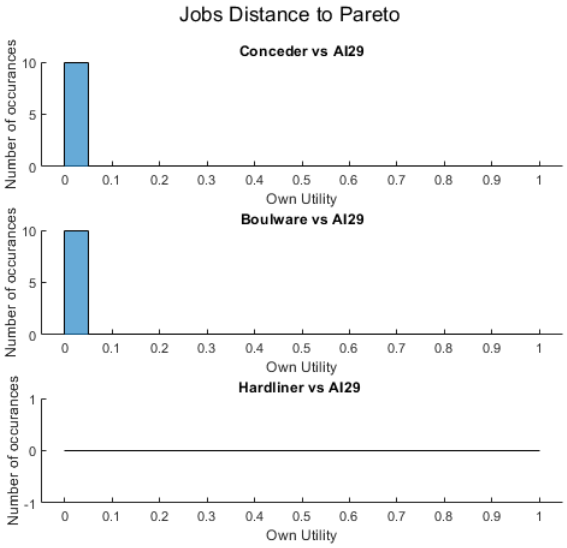

Figure 9: Histogram of the distance to the Pareto frontier sorted on Agents competing.

optimal to send any value higher then that value in that domain. It stands to reason that this is only possible if the domains are independent of each other.

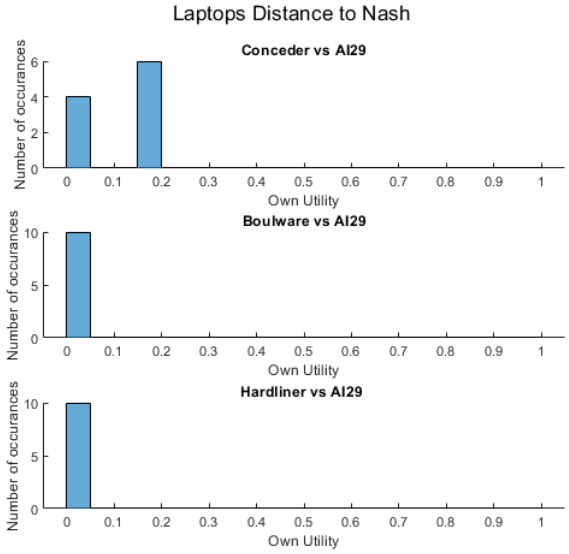

Figure 10: Histogram of the Nash product sorted on Agents competing.

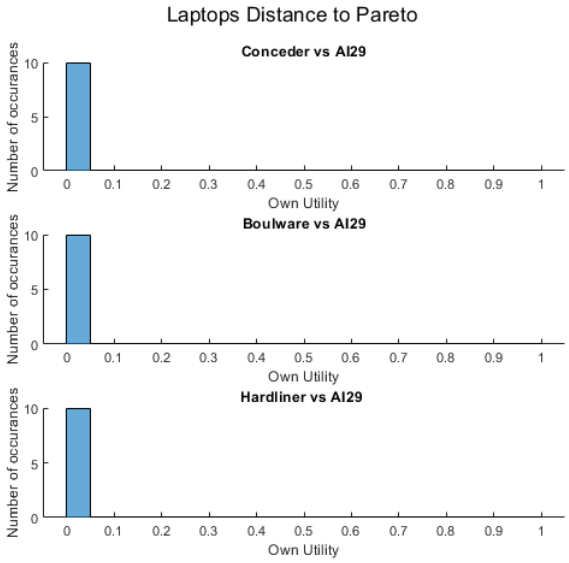

Figure 11: Histogram of the distance to the Pareto frontier sorted on Agents competing.

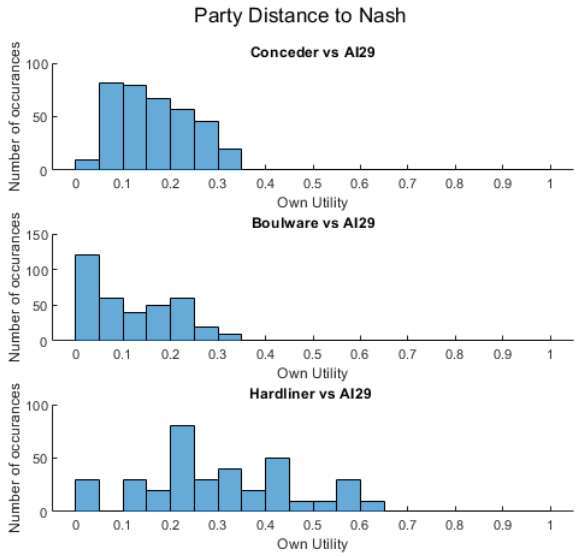

Figure 12: Histogram of the Nash product sorted on Agents competing.

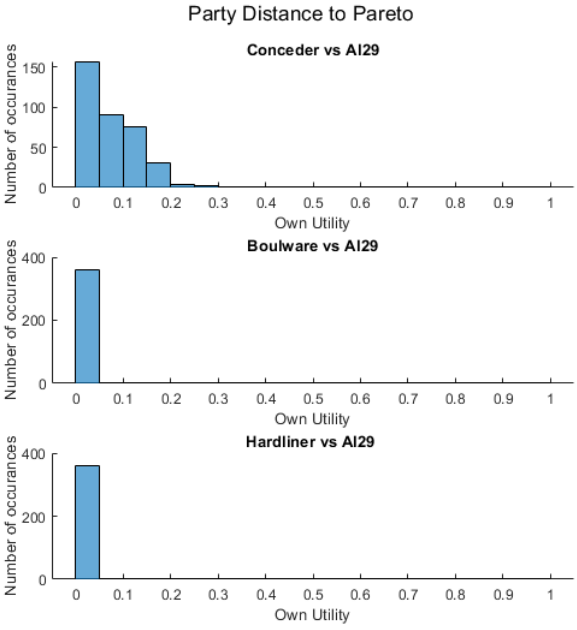

Figure 13: Histogram of the distance to the Pareto frontier sorted on Agents competing.

5: Possible improvements

We have previously seen that automated negotiation on behalf of humans have been developed to improve overall outcomes of the negotiation process. The main directions of improvements being offered by automated negotiation is to reduce the limited rationality of humans by modelling utilities. The available actions for parties consist of terminating the negotiation, accepting a received bid, or generate a new bid. The latter is the only active form of communication between the agents: received bids provide information about the opponents utilities and the agents strategy.

In order to achieve a more open communication with the aim to improve the overall utility of the deal, parties might be extended with different capabilities. These capabilities could be anything along the range of:

- disclosing part of the utilities in order to find resemblances in the utility profiles;
- providing a 'dealbreaker' issue value within a specific domain;
- present (part of) your strategy at the beginning of the negotiation (interesting for eg hardliner agent: 'I will not make concessions so just take what you get now');

Any of these examples of providing more information resemble possible actions that humans might take in the process of negotiation. This conflicts with the aim of automated negotiation to improve rationality in the negotiation setting.

Therefore, if both parties are capable of taking any additional action that provides some sort of information to the opponent, the opponent might choose to ignore this capability and use the additional received information to improve its opponents model. This will result in a stronger position for the opponent. Hence, both parties will end in an equilibrium where neither one will make use of the additional capabilities.

Another recommendation is to test which are the most optimal threshold values for the starting player and the second player. These values are now set at 0.9 and 0.7 respectively. Tests could be run where these values are changed to analyze the average utility against multiple parties to find the most optimal values for the thresholds.

4 CONCLUSION

In this reported we have introduced an agent to be used in automated negotiation. The implementation of our agent uses a different strategy for the starting turn in the negotiation session. Where starting uses a threshold ≥ 0.9 where bids are drawn based on a Nash product list. Starting second has an acceptance threshold of 0.7 which reduces over time. The agent is able to attain high utilities in the tested domains. If the negotiation session is against an agent which is willing to find an outcome with a deal we often end up with a high utility. An utility domain where there are more possible outcomes as there are more issues in the domain allow for a better negotiation session. This is evidenced by the fact that the utility range on the party domain is greater than that of the laptop and the job domain.

4.1 Discussion

While the results of agents are a good indication that this is a well performing agent, the question is: can it do our negotiations for us? At this point in time, this is not yet the case. There would have to be made some improvements. One of these improvements would be to improve the preference uncertainty. In this scenario the opponent has come up with a bid that is not in your preference list, but is actually acceptable for the human. The negotiation agent will not take this into account and reject the offer. This is an indication that the agent does generally not know exactly what the human wants, this can maybe solved by letting the human control parts of the agent during the negotiation session.

4.2 Evaluation

Negotiating over party resonates with us as students, although the authors can't appreciate negotiating with our own agent for a party due to it's selfish goal of utility. As a group we strived for a good end product, where we cooperated the feedback to improve our work. The structured manual allowed for a direct research approach regarding the creation of the agent. The teamwork was on point and was often preformed in dedicated session on location. We thank T. Baarslag and the other coder for the template code. We appreciate the insight the lectures gave.

REFERENCES

[1] T. Baarslag, W. Pasman, K. Hindriks, and D. Tykhonov. 2019. Using the Genius Framework for Running Autonomous Negotiating Parties.
[2] Basar, Tamer, Olsder, and Geert Jan. 1999. *Dynamic noncooperative game theory*. Vol. 23. Siam.
[3] Liu Kexing. 2011. A survey of agent based automated negotiation. In *2011 International Conference on Network Computing and Information Security*, Vol. 2. IEEE, 24–27.
[4] Morrow and James D. 1994. *Game theory for political scientists*. Vol. 23. Princeton University Press.

Appendices

On the next pages a few phase profiles of various PF elements are given.

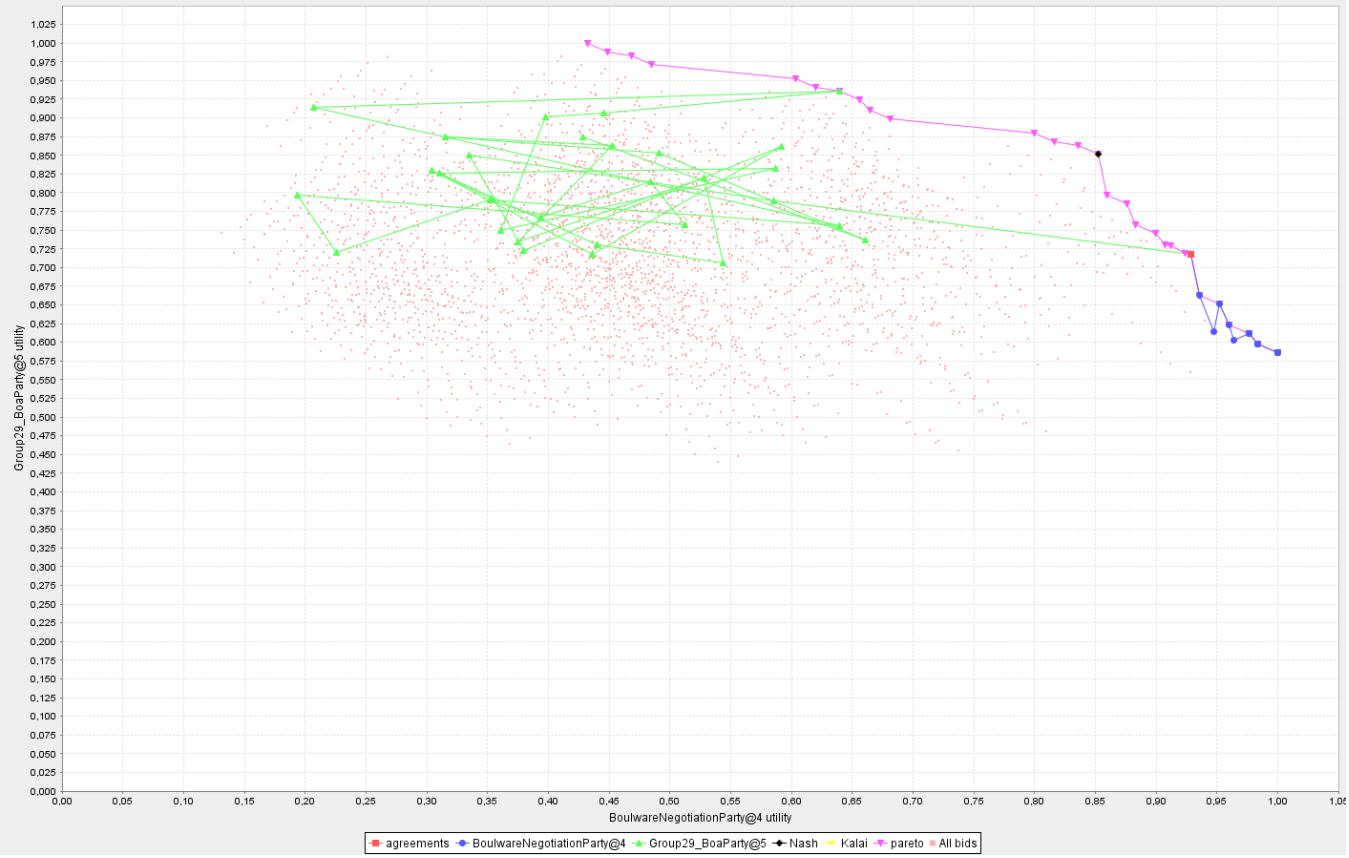

Figure 14: Boulware vs AI29

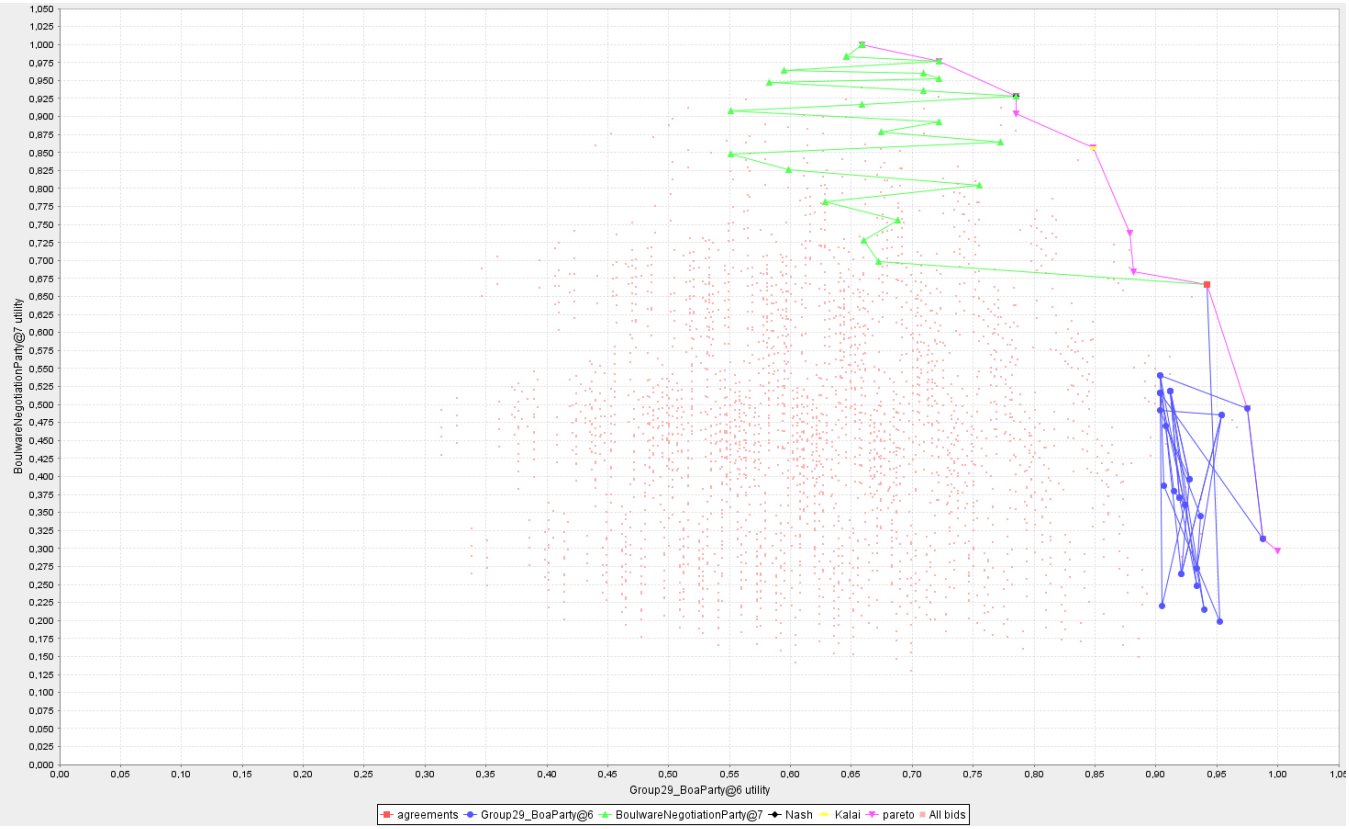

Figure 15: AI29 vs Boulware

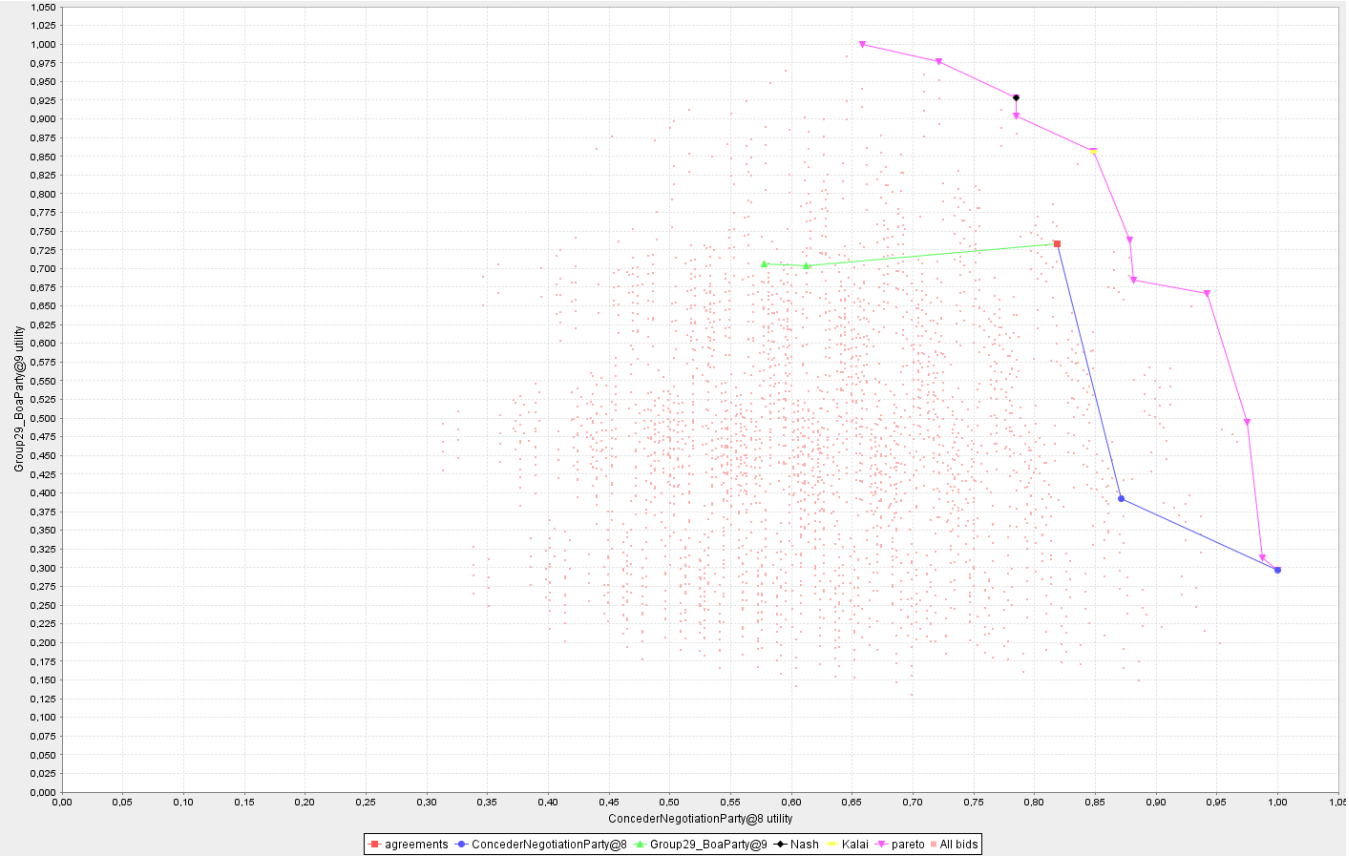

Figure 16: Conceder vs AI29

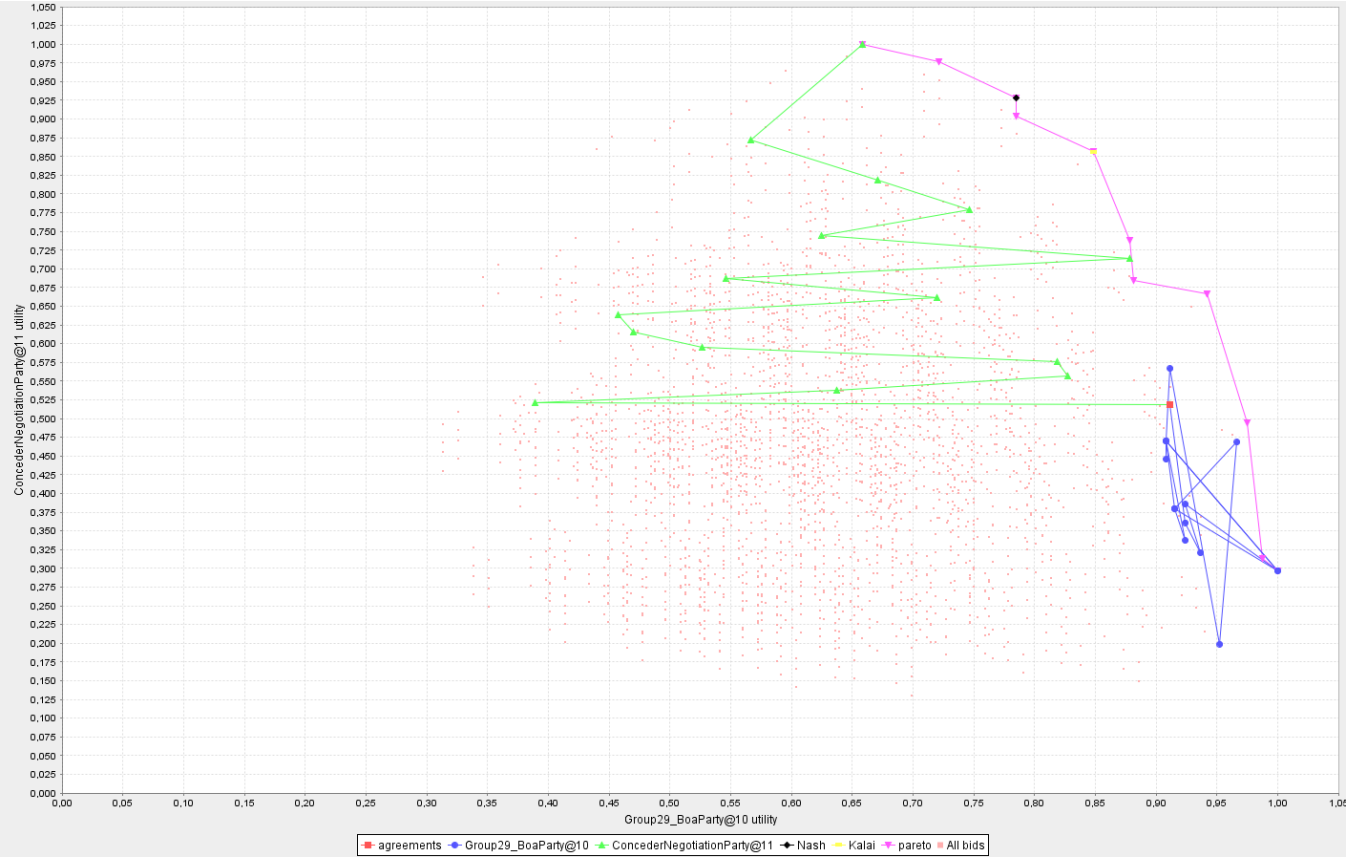

Figure 17: AI29 vs Conceder

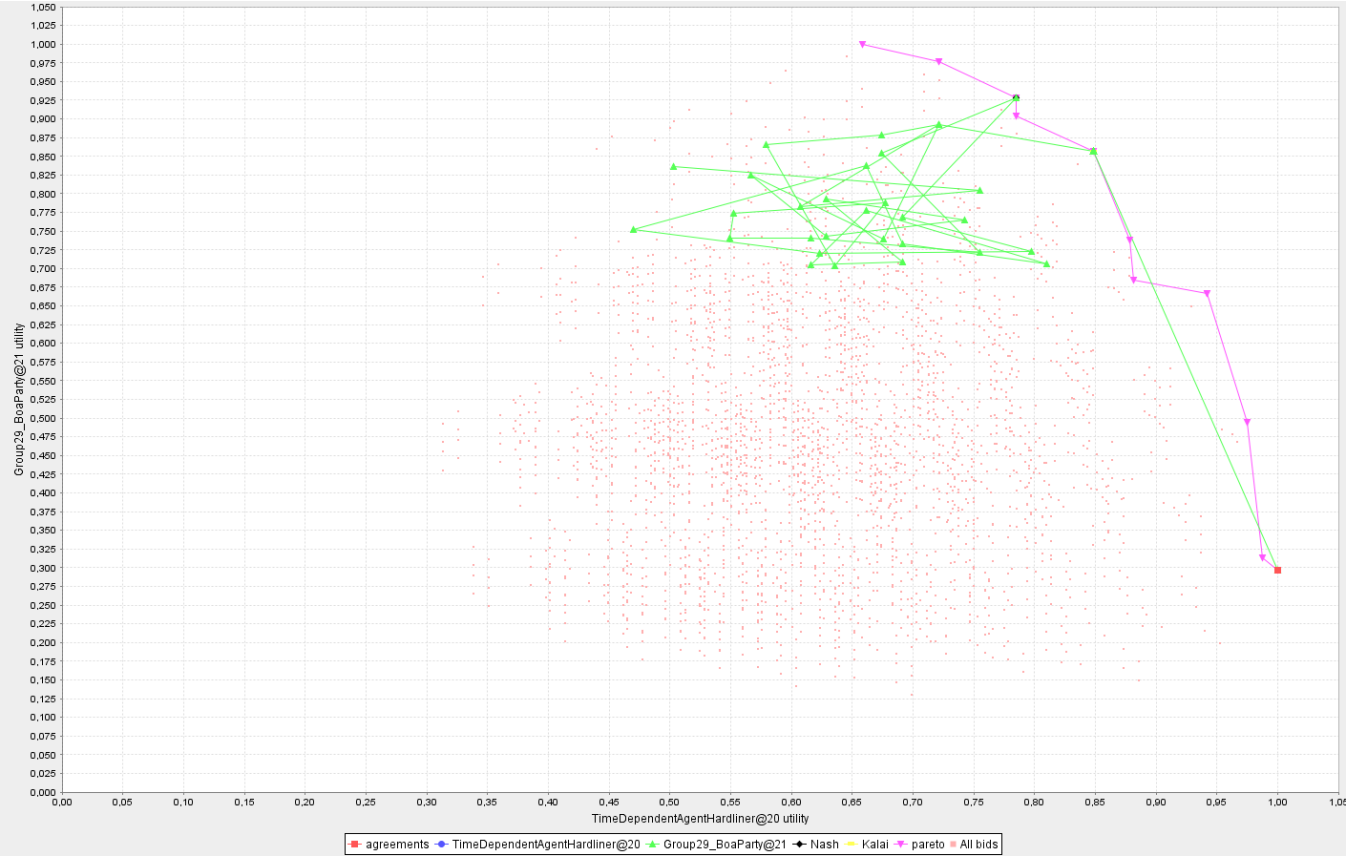

Figure 18: Hardliner vs AI29

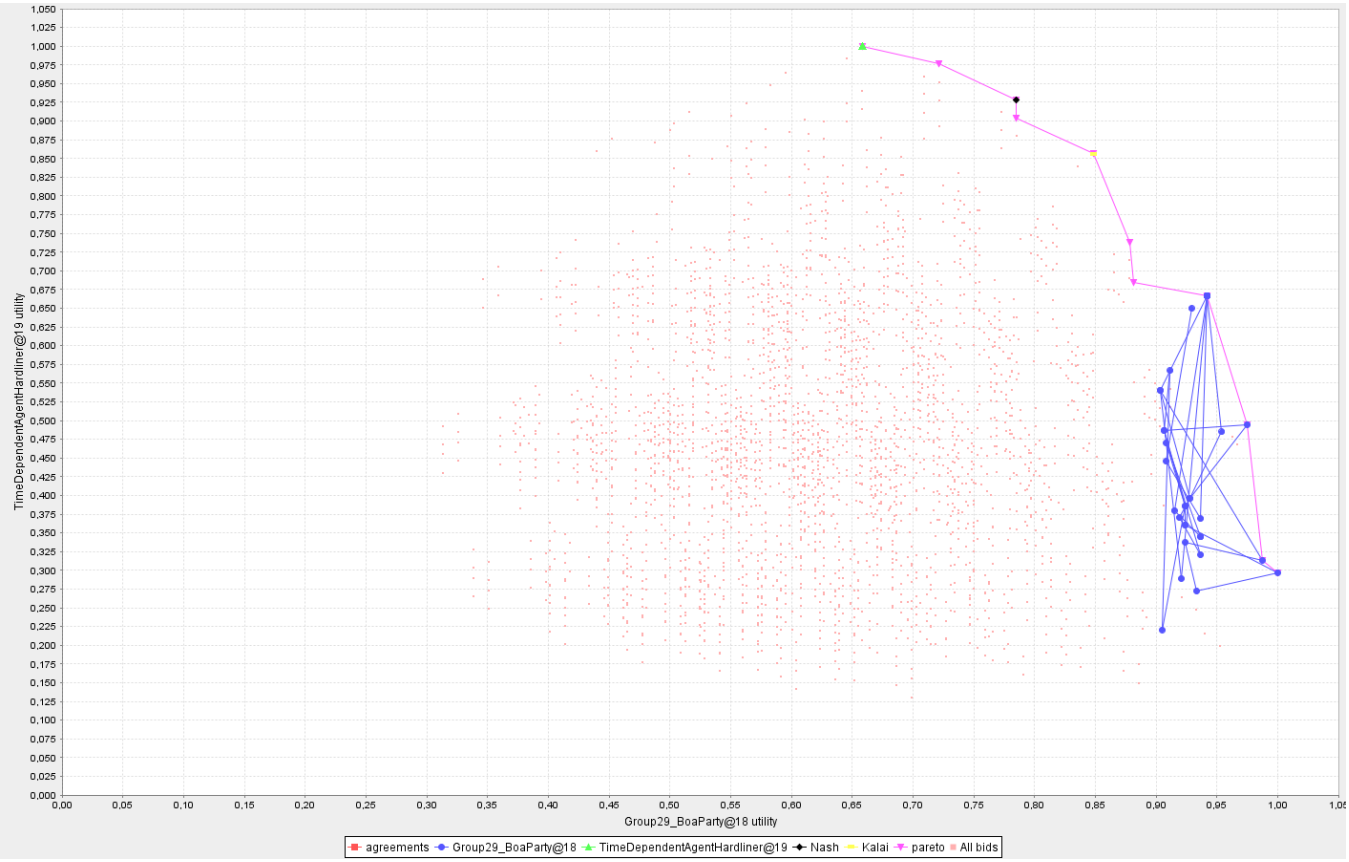

Figure 19: AI29 vs Hardliner

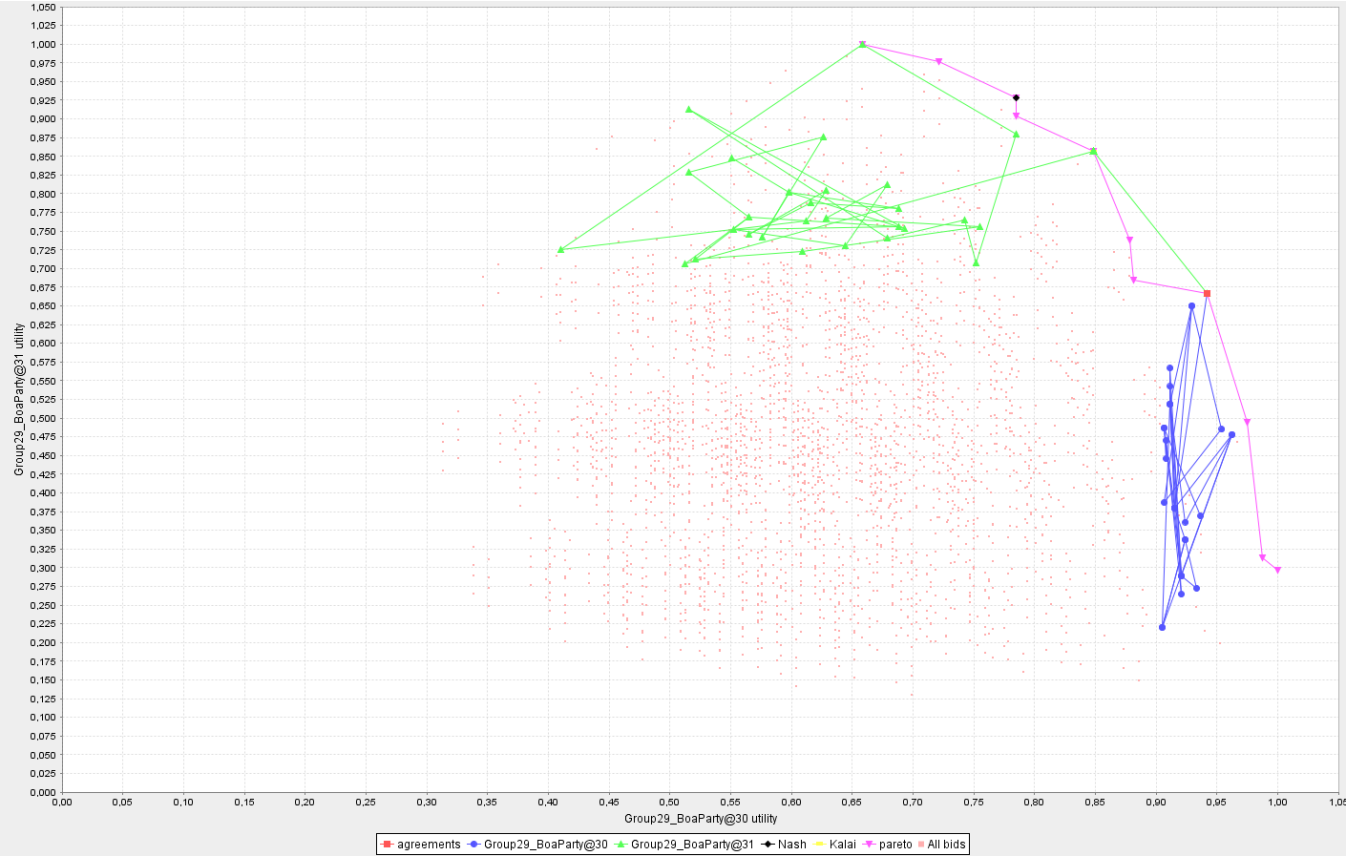

Figure 20: AI29 vs AI29 Party utility 1-2

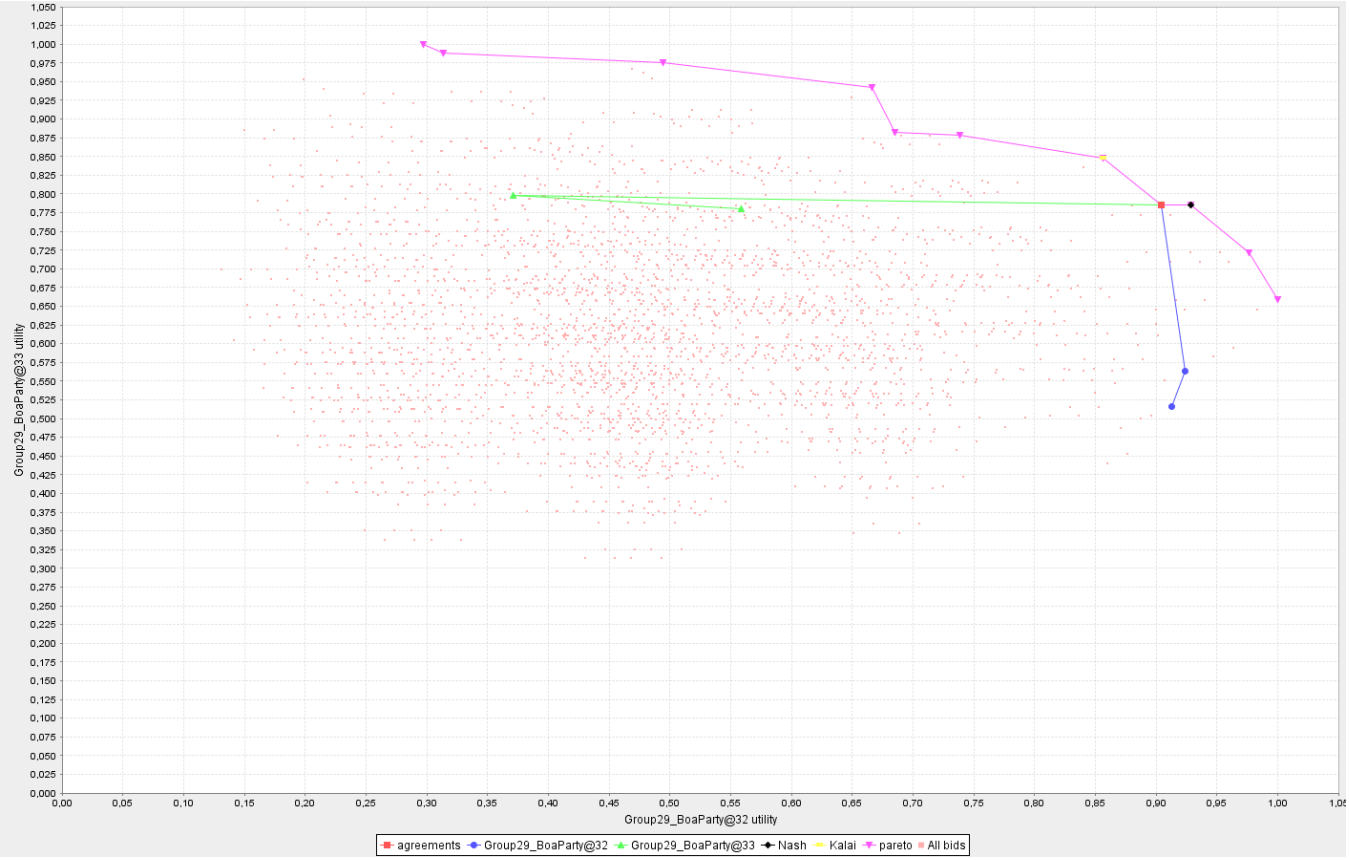

Figure 21: AI29 vs AI29 Party utility 2-1

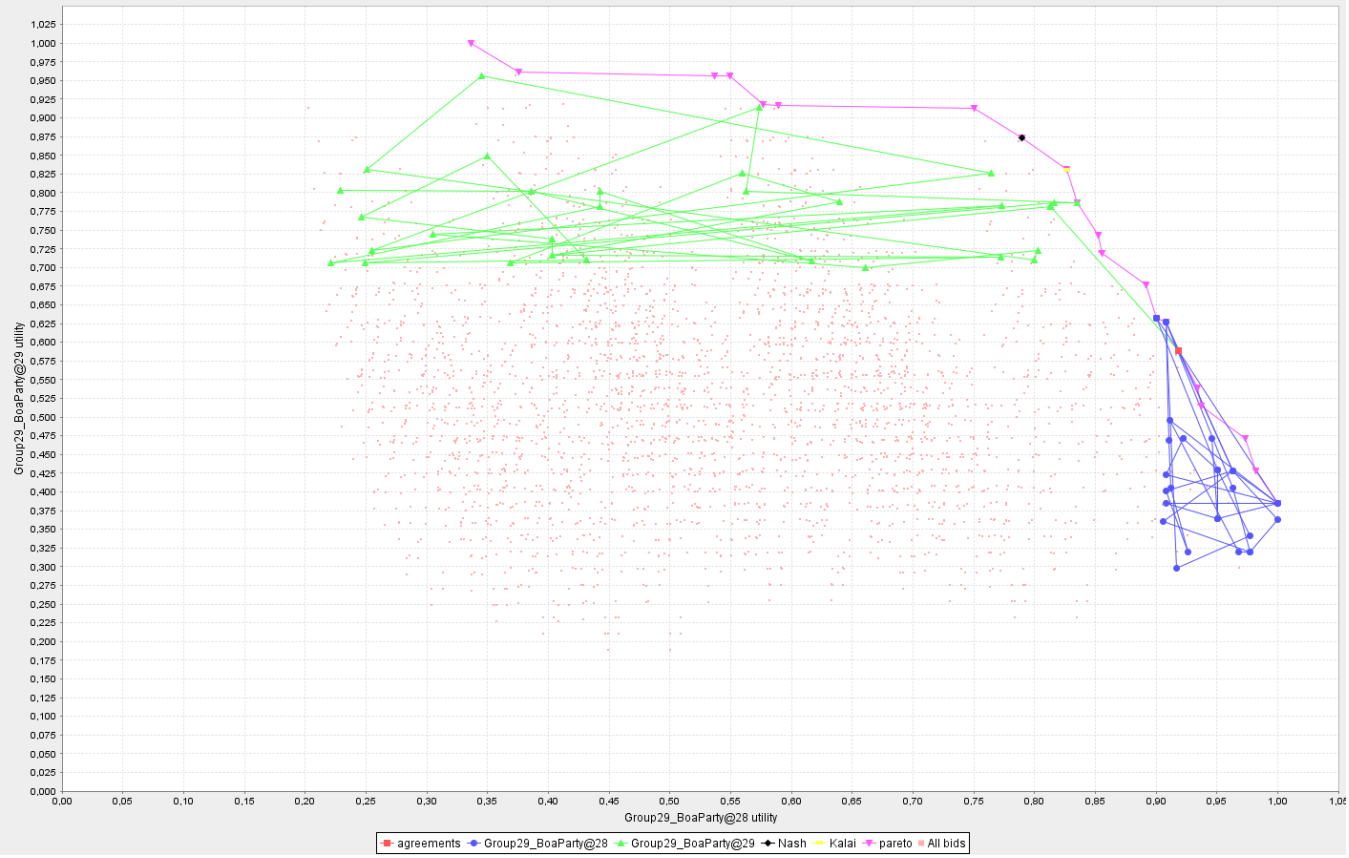

Figure 22: AI29 vs AI29 Party utility 7-8