AntiAgent: a Negotiation Agent for ANL2024

Panagiotis Aronis (p.aronis@students.uu.nl)
Sander van Bennekom (s.vanbennekom1@students.uu.nl)
Mats Buis (m.p.buis@students.uu.nl)
Vedant Puneet Singh (v.p.singh@students.uu.nl)
Collin de Wit (c.r.dewit@students.uu.nl)

Department of Information and Computing Sciences
Utrecht University
Netherlands
April 15, 2024

1 Introduction

This report presents *AntiAgent*, a negotiating agent submitted to the *Automated Negotiating Agent Competition (ANAC)* for the *Automated Negotiation League (ANL2024)* challenge. Competitor agents participate in a bilateral negotiation tournament over multiple negotiation domains against each other using the *Alternating Offers Protocol (AOP)*. Negotiators exchange offers according to their conflicting interests represented by their respective *utility functions*. The other party can respond with an acceptance, a counteroffer, or choose to (prematurely) end the negotiation. The goal is to reach a mutually acceptable agreement before the negotiation deadline where each agent achieves the agreed outcome's utility for them. Otherwise, if the deadline is reached with no agreement or the negotiation ends early for any reason agents receive their respective *reservation values*.

For this year's challenge, *ANL2024*, negotiating agents have access to the opponent's preference profile, having complete knowledge of their utility function. This makes the negotiation domains essentially single issue. However, the opponent's reservation value is private information so there is still a lot of uncertainty during the negotiation sessions. The winner of the challenge is the agent that scores best on *individual utility*. Unlike previous competitions, agents cannot learn between different negotiation sessions and there is no discount factor for utilities. A significant part of the challenge is to get an accurate estimate of the opponent's reservation value during the negotiation session and exploit this knowledge for own advantage.

The implementation of *AntiAgent* follows the *BOA* agent framework ([1], [2]). According to this, the negotiation strategy is divided into three interacting components, namely, the *bidding strategy*, which generates the bids the agent offers to the opponent, the *acceptance strategy*, which decides if the agent should agree to a proposed offer, and the opponent model, or to be exact, the *reservation value model* in our case, which attempts to learn the opponent's private information based on their behavior. For building, testing and evaluating the agent the *NegMAS* platform was used with *Python*.

2 Agent Description

The submitted negotiating agent, *AntiAgent*, is named appropriately as it generates the optimal response to an assumed opponent acceptance strategy. The core strategy element is a *expected utility* calculation depending on the number of bidding rounds remaining. Our work is heavily inspired by the analysis in [3]. This paper demonstrates how to make optimal concessions against specific classes of acceptance strategies. We make use of the general framework of sequential decision techniques to maximize the expected utility with our bidding curve. We add on this work in some significant ways given our particular negotiation challenge. First, we leverage the knowledge of the opponent's utility function to apply the expected utility computation to non *zero-sum* scenarios. Second, the bidding strategy was made adaptive to the behavior of the opponent. In particular, we make use of an opponent reservation value model, partly based on the *NegMAS* default negotiator *RVFitter*, to get an estimated distribution of the opponent's reservation value. Then we use this estimate to dynamically update our assumed opponent acceptance model which in turn changes our optimal response based on expected utility. Finally, we attempt to model opponent acceptance strategies other than *satisficing*, that is always accepting anything more than their reservation value. However, modeling the opponent as satisficing turned out to be a surprisingly tough benchmark to beat as we will discuss later.

The above optimal response strategy based on expected utility gained renders *AntiAgent* an incredibly competitive negotiator. Due to the expected utility monotonically increasing with more bidding rounds to go, the agent tends to not concede at all for the most part of the negotiation, a tactic that generally works given the success of past *ANAC* competitors like *HardHeaded* [4]. However, with fewer rounds remaining it is able to make timely concessions achieving enough agreements in the end. Because there is no discount factor for utilities in this challenge, it is always a good idea to be patient, and our agent takes this to the extreme getting most agreements on the very last negotiation round exactly as planned with the optimal bidding curve. The agent's acceptance strategy also fully relies on the expected utility calculation, where the expected utility for a turn is always lower than our next optimal bid. The previous two strategy components are made dynamic with the use of an opponent reservation value model that employs two different kind of reservation value estimation, a simple one based on the opponent negotiation gap and a more sophisticated one based on best curve fitting assuming that the opponent's bidding strategy resembles a *time-dependent tactic*.

3 Agent Implementation

The *AntiAgent* strategy implementation makes use of the *NegMAS* platform in *Python*. In particular, the negotiating agent is a python class extending *negmas.sao.SAONegotiator*. The function *on_preferences_changed* is used for the agent initialization (for *ANL2024*, only called once before starting each negotiation session). The initial values for important class variables will be given in the relevant subsections below. All other functionality is captured in the *__call__* class method and follows the *BOA* agent framework. Each call to the agent object implements one round of the *AOP* protocol taking the current negotiation state (*SAOState*) and returning an appropriate type of response (*SAOResponse*). First, the *update_partner_reserved_value* function updates our current best estimate for the opponent's reservation value. This estimate is then used in *opponent_acceptance_probability* function which models the likelihood of the opponent accepting a particular offered outcome. The latter function is an integral part of the *maximize_expected_utility* function that generates the optimal bidding strategy and the accompanied expected utility for each remaining round of the negotiation session (of course we need to recalculate this trajectory when the estimate of the opponent's reservation value changes). With this calculation done, the *acceptance_strategy* first determines the acceptability of the opponent's offer by comparing it with our current expected utility threshold and then the *bidding_strategy* determines the counter offer using the current optimal outcome that achieves the expected utility.

Before delving into the specific details of each of these functions we should mention some important changes compared to our original envisioned negotiation strategy components. First and foremost the original (multi-phase) time-dependent tactic was scraped for the current expected utility module that fully determines both our current bidding and acceptance strategies. Even though with enough bidding rounds remaining the current bidding curve somewhat resembles an extreme *Boulware* tactic, the current version enjoys many advantages. In addition to being more unpredictable and generating more accurate timely concessions closer to the negotiation deadline, it is a formulation of an optimal response to the assumed opponent acceptance model that guarantees a very competitive negotiator (given that the assumption is justified) and it is also more easily adaptive to the opponent behavior benefiting from the opponent reservation value model. The second, less important, change concerns the reservation value model. Although our agent implementation still uses the envisioned *curve fitting* technique inspired by the default *RVFitter* negotiator, we also employ a second more simplistic approach to reservation value modeling. Curve fitting assumes that the opponent bidding history is the result of following a time-dependent tactic curve and attempts to find the best reservation value and concession exponent that fit the offered outcomes so far. Even though this technique is extremely potent against time-dependent tactics it can be highly inaccurate for other bidding strategies and it is also quite expensive to perform every single round. A solution that solves both of these issues is to employ a simpler rough estimate of the opponent's reservation value based on the current negotiation gap for the most part of the negotiation session and only switch to curve fitting closer to the deadline where we can get more accurate prediction due to more data points being available, exactly when it really matters as the latter stage of a negotiation session is far more crucial to achieve good performance.

3.1 Update Partner Reserved Value

As we already mentioned this strategy module employs two different methods of very different sophistication to balance out speed and generality in the early negotiation stages with increased accuracy to squeeze out slightly better deals closer to the negotiation deadline. The process starts by always keeping track of the opponent's new offer, thus saving their whole bidding history in the form of (relative time) timestamps, individual utilities of outcomes and the minimum and maximum of the latter. Then we employ our learning techniques depending on the relative time of the negotiation session. For the first 90% of the negotiation session we just check the biggest own concession the opponent has made so far and, assuming that they offer no outcomes below their reservation value, we set our current reservation value estimate to half that value. That is, we starting by estimating the reservation value at 0.5 and this estimate only goes down during this stage proportionally to the greatest concession made from the opponent. For the last 10% of the negotiation session we employ the more sophisticated curve fitting technique. This method assumes (in addition) that the opponent follows a time-dependent tactic for bidding. In particular, a *polynomial aspiration curve* is used like the one that is used by the default *NegMAS* negotiators, *Boulware*, *Linear* and *Conceder*. Then the *scipy.optimize.curve_fit* method is used to estimate the best reservation value and concession exponent that fit the opponent's bidding history so far. The reservation value estimate is updated only if there are at least 25 unique outcomes offered from the opponent in order to avoid overestimating the reservation values of opponents that have not conceded much yet.

3.2 Opponent Acceptance Probability

After updating the opponent reservation value estimate and in order to be able to compute our new optimal expected utility we need to update our opponent acceptance strategy model, that is, what is the probability that the opponent will accept an outcome with certain utility for themselves. Initially, a static acceptance model was used as a benchmark that was really similar to the work in [3]. The prior knowledge of the opponent reservation value was just assumed to be a uniform distribution over all possible values and we did not update this knowledge according to the opponent's behavior at all. In addition, the opponent acceptance strategy was modeled to be *satisficing*, that is, we assume they will always accept anything that is better than their real reservation value. This way the probability that the opponent accepts an outcome of individual utility u is $p(u) = u$. The resulting agent with no reservation value modeling and completely non-adaptive bidding and accepting strategies was already outperforming all default *NegMAS* negotiators with time-dependent tactics. Now to make use of our reservation value estimates we can assume that the opponent reservation value follows a *normal distribution* with mean equal to our current best reservation value estimate. The standard deviation is also dynamically computed so the normal distribution fits exactly in the range of all assumed possible opponent reservation values (the greatest concession of the opponent so far is exactly 6 standard deviations), so the uncertainty about the reservation value reduces over time. Assuming that the opponent is satisficing as before, we get a *logistic function* acceptance probability response for the opponent, $p(u) = \Phi((u - \mu)/\sigma) = 0.5 + 0.5erf((u - \mu)/(\sigma\sqrt{2}))$. Significant attempts were made to outperform this last version of the agent. One of the most promising alternative opponent acceptance strategies made use of the curve fitting model estimation of the opponent concession exponent and assumed that the acceptance strategy mirrors the bidding strategy as modeled by the *RVFitter* module's aspiration curve. This agent version was perfect against time-dependent tactics but it was not robust enough to generalize across other opponents. The surprisingly hard to beat satisficing opponent model demonstrates that against rational opponents it always comes down to satisficing behavior when close to the negotiation deadline when most of the agreements actually take place.

3.3 Maximize Expected Utility

Having a fully specified adaptive opponent acceptance strategy that makes use of our opponent reservation value estimates we are now ready to exactly calculate the optimal response to this acceptance strategy. That is to say, what is the optimal bidding curve for the remainder of the negotiation session that maximizes our expected utility, only considering our bids and the (assumed) probability that the opponent will accept any of them. The computation of *expected utilities* and *optimal bids* for each of the remaining negotiation (bidding) rounds works by backwards induction from the negotiation deadline. First, the agent gets the number, n, of own bidding rounds it has left. Important detail is that because of the asymmetry of the *AOP* negotiation protocol the agent to bid first always gets an extra bidding round compared to their opponent no matter the negotiation deadline. For no remaining bidding rounds, $n = 0$, there is no bid we can make, so there is no opportunity for the opponent to accept and (ignoring opponent's offers) the best (and only) thing we can do is to end with an unsuccessful negotiation getting our reservation value, rv. Then we have that the expected utility with zero bidding rounds remaining is $U_0 = rv$. Can we do any better for $n = 1$? We have exactly one bid to make, offering some outcome ω with own utility u_A and opponent utility u_B. For each outcome we have modeled the acceptance probability of the opponent so we can compute the expected utility we achieve for each possible offered outcome. Doing the math we have that the best outcome to offer is the one that *maximizes* the *extra utility gain* for this round, $U_+ = (u_A - rv) * p(u_B)$, and our new optimal expected utility is $U_1 = U_0 + U_+$. We can see that as a kind of *Nash point* solution adjusted by the assumed opponent acceptance model. This relation of course generalizes to every round and so we have: $U_n = U_{n-1} + argmax_\omega(u_A(\omega) - U_{n-1}) * p(u_B(\omega))$. Intuitively, the expected utility with one less round is assumed as our new 'reservation value' and we try to maximize the extra utility gain on top of that if possible. Some properties about this process should be noted. The modeled opponent's acceptance probability is a strictly increasing function, therefore the quantities to be maximized are always larger at the *pareto frontier* points. This means that our agent needs to only consider *pareto-optimal* outcomes for bidding and we filter through these during initialization using *negmas.preferences.pareto_frontier*. Another property that emerges due to the same reason is that the bidding history generated is (not strictly) decreasing which essentially means that our expected utility with more rounds remaining is always higher.

3.4 Acceptance Strategy

With the expected utility computation done, most of the work for our acceptance strategy is already done. We already know our current expected utility so it is reasonable to accept an opponent's offer that gives us an individual utility higher than this threshold. However, because we recompute the expected utilities when our opponent reservation value estimate is updated, it is possible that after a sudden concession from the opponent, the current expected utility could be higher than the outcome we offered the previous turn (we think that we already conceded more than enough). In this case we lower our acceptance threshold to our last utility offered. Due to the acceptance thresholds being derived via maximizing expected utility, it is evident that both negotiation time and opponent's behavior greatly influence the decision made by the acceptance strategy. Finally, when rejecting an offer, a counteroffer is always made as there is never an incentive to prematurely end the negotiation due to the absence of any utility discount factor.

3.5 Bidding Strategy

Likewise for the bidding strategy almost all of the work is already done by calculating the optimal bids as response to the opponent modeled acceptance strategy. The only thing the bidding strategy should decide is when to go with the computed optimal offer or just repeat our last offer. Even though all bidding curves computed with expected utility maximization are monotonic, it is still possible as we mentioned above for the current optimal bid to have higher utility for us compared to our previous offer. In this case we ignore the optimal offer and repeat our previous offer again, thus ensuring that our bidding history is (not strictly) monotonic. This is a deliberate design choice in an attempt to render *AntiAgent* a bit more clear and friendly to behavioral strategies that could hold a 'grudge' if we moved back from a previous offer. All optimal offers are generated by maximizing the expected utility recurring formula and our first offer is no different. When the number of pareto optimal points is less than the remaining biding rounds till the negotiation deadline we will indeed offer our best outcome due to the monotonicity of expected utility. Otherwise, it is perfectly possible for the first bid to already be conceding if this is needed to maximize expected utility. From our analysis it is evident that both agents' preference profiles and opponent's behavior via the adaptive opponent acceptance model is taken into account when generating the optimal bids to offer.

References

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

[2] Tim Baarslag et al. "The Significance of Bidding, Accepting and Opponent Modeling in Automated Negotiation." In: *ECAI*. 2014, pp. 27–32.

[3] Tim Baarslag et al. "Optimal non-adaptive concession strategies with incomplete information". In: *Recent Advances in Agent-based Complex Automated Negotiation* (2016), pp. 39–54.

[4] Takayuki Ito et al. *Complex automated negotiations: Theories, models, and software competitions*. Springer, 2013.