

Design and Evaluation of a Negotiation Agent with Opponent Utility Estimation in the Automated Negotiation League

Kota Fujimoto
Nagoya Institute of Technology

1 Introduction

The Automated Negotiation League (ANL) is a competition in which agents negotiate under private utility functions and limited negotiation steps. An ANL agent must not only obtain high utility for itself, but also estimate the opponent’s utility function and run robustly over many negotiations without exceptions. In ANL 2026, opponents and domains are not fixed in advance. Therefore, an agent that is overfitted to a specific opponent is less useful than one that remains stable across diverse utility distributions. In addition, the final ranking is strongly affected not only by the average score, but also by how often the agent falls into low-score negotiations. For this reason, the mean score, median, quartiles, interquartile range, and running time should be interpreted together.

This paper reports the design and evaluation of MajiKayoAgent, our submitted agent for ANL 2026. We initially investigated utility-function obfuscation, in which a virtual utility function changes the visible offer sequence and makes opponent-side inference more difficult. However, preliminary experiments showed that such perturbation can reduce the agreement rate. The final submitted agent therefore uses real-utility-based monotonic concession as its main behavior and uses opponent utility estimation only for late-stage offer selection.

The purpose of this paper is not only to describe the implementation of MajiKayoAgent, but also to explain how the design choices appear in the tournament results. We focus on three choices: preserving self-utility in the early stage, using opponent utility estimation only after enough observations are collected, and excluding utility-function obfuscation from the final submitted version.

2 Design Challenges in ANL

In ANL negotiations, an agent observes its own utility function but cannot directly observe the opponent’s utility function. Therefore, it must infer the opponent’s preferences from the offer history and select outcomes that are likely to be acceptable. However, this creates a trade-off. If the agent follows the estimated opponent utility too strongly, it may lose too much self-utility. If it prioritizes only self-utility, it may continue making offers that the opponent will not accept, leading to disagreement or low scores.

Opponent utility estimation itself is also difficult. First, an opponent’s offer sequence does not necessarily reveal its true utility ranking. The opponent may also be strategic and may make offers that hide information. Second, the number of negotiation steps is limited, so the agent often has to make acceptance and concession decisions before enough observations are available. Thus, an ANL agent must improve estimation accuracy while still behaving safely under uncertainty.

In this work, we treat the problem as selecting from a self-utility-safe candidate set. Opponent utility estimation is not used to choose from all outcomes. Instead, the agent first

restricts candidates to high self-utility outcomes and then uses the opponent model only to rank those candidates. This limits the damage caused by estimation errors.

3 Design of MajiKayoAgent

MajiKayoAgent combines a self-utility-oriented offer strategy with an opponent utility model built from the opponent’s offer history. Its design has three main components.

First, at the beginning of a negotiation, the agent offers the outcome with the maximum self-utility twice. This avoids early unfavorable agreements and gives the agent time to observe the opponent’s responses and counter-offers. Second, from the third offer onward, it selects an unused outcome that is above the reservation value and has lower self-utility than the previous self offer. This prevents repeated offers and creates a concession sequence that is visible to the opponent. Third, after the negotiation progress exceeds 0.5, the agent takes the top four candidates by self-utility and offers the one with the highest estimated opponent utility. This keeps the agent from over-conceding while increasing the probability of agreement in the late stage.

Candidate outcomes are enumerated from the outcome space and sorted by the agent’s own utility. Self-utility is normalized to the range from 0 to 1 using the minimum and maximum utility values in the domain. This normalization allows the same decision rules to be applied even when utility scales differ across domains. Outcomes that have already been offered are generally not reused. This avoids wasting steps and also makes the concession process visible to the opponent.

3.1 Opponent Utility Model

MajiKayoAgent estimates the opponent’s utility from the opponent’s offers. At initialization, the model assigns neutral values to outcomes. It then updates the estimated values based on the issue values that appear in the opponent’s offers and on the timing of those offers. When a self offer is rejected, the model also treats that outcome as less attractive to the opponent.

The model is intentionally not used too aggressively. In the early stage, only a small number of opponent offers have been observed, so the estimated ranking can be unstable. MajiKayoAgent therefore prioritizes self-utility in the first half and uses the opponent model only after the negotiation progress exceeds 0.5. Even then, the model is applied only to a short list of high-self-utility candidates. This design localizes the effect of estimation errors.

The acceptance strategy compares the current opponent offer with the agent’s next planned offer. If the normalized self-utility of the opponent offer is at least as high as that of the next self offer, the agent accepts it; otherwise, it rejects it. Thus, the acceptance threshold is not fixed, but changes according to the agent’s planned concession path. For example, if the next planned self offer is already a significant concession, an opponent offer with comparable self-utility is

worth accepting. Conversely, when the agent can still make high-self-utility offers, it should not accept a low opponent offer. This ties the acceptance policy and offer policy to the same utility scale.

3.2 Negotiation Procedure

During a negotiation, MajiKayoAgent follows a simple procedure. When preference information becomes available, it enumerates all candidate outcomes and computes their self-utility and normalized self-utility. It then sorts the candidates by self-utility and initializes a set for tracking already offered outcomes. In the proposal phase, the agent selects the maximum-self-utility outcome in the early stage. After that, it searches for candidates whose self-utility is lower than that of the previous self offer. In the late stage, it evaluates the estimated opponent utility of the top four candidates and chooses the candidate that is estimated to be best for the opponent.

In the response phase, the agent first updates the opponent utility model using the current opponent offer. It then computes the offer it would make next under the same state and compares the opponent offer with that planned self offer. This is equivalent to comparing the utility obtained by accepting now with the utility of rejecting and continuing with the next offer. Therefore, the decision can adapt to the current concession state instead of relying on a fixed acceptance threshold.

The advantage of this procedure is that the strategy is relatively simple and interpretable. Its disadvantage is computational cost. Because the agent enumerates outcomes and scans candidate sets repeatedly, the response time can grow. Improving the final rank therefore requires both better strategy design and lower implementation cost.

4 Preliminary Evaluation

Before submission, we compared offer strategies, acceptance strategies, and opponent utility models using mixed scenarios close to the ANL platform distribution. The evaluation used 20 scenarios, 80 outcomes per scenario, and 60 negotiation steps. The results showed that non-repeating offer strategies that prioritize self-utility tend to obtain high self-utility, while the accuracy of opponent utility estimation is affected by the regularity of the offer sequence. The evaluation considered agreement rate, self-utility, score, opponent utility estimation accuracy, and the number of steps. If only self-utility is examined, strategies that avoid concession appear attractive. However, if the agreement rate is low, the final score does not improve. Conversely, even a highly accurate opponent model can reduce self-utility if its estimates are used too strongly for offer selection. Therefore, we evaluated strategies by considering multiple metrics rather than optimizing a single one. The compared strategies included linearly decreasing threshold strategies, random candidate selection, filtering based on estimated opponent utility, and utility-function obfuscation that changes the offer order through a virtual utility function. Obfuscation was promising in the sense that it could make the visible offer sequence harder to infer from, but it weakened the relation between real utility and the offer order. As a result, it also made acceptance decisions more difficult.

For the final configuration of MajiKayoAgent, a local evaluation over 80 negotiations produced an agreement rate of 93.75% and a normalized Kendall’s tau of 0.775 for opponent utility estimation. The agreement rate was 100% in the Amsterdam, Berlin, and Camera scenarios, but dropped

to 75% in the Laptop scenario. This suggests that a real-utility-oriented concession sequence alone is not always sufficient to approach the agreement region, depending on the opponent’s reservation value and preference structure. On the other hand, in scenarios with high agreement rates, the same design was able to reach agreements while preserving self-utility. Therefore, for the final submission, we prioritized safe concession based on real utility over strong perturbation using a virtual utility function. This decision is also reflected in the tournament result, where the agent produced no exceptions.

Table 1 shows the scenario-level local evaluation of MajiKayoAgent. The score includes negotiation performance and opponent utility estimation accuracy, and tau is the normalized Kendall’s tau. The agreement rate was 100% in Amsterdam, Berlin, and Camera, but dropped to 75% in Laptop. The average number of steps was also much larger in Laptop, suggesting that the agent often needed a long search before reaching an acceptable offer.

Table 1: Scenario-level local evaluation of MajiKayoAgent

Scenario	Score	Tau	Agreement	Steps
Amsterdam	0.924	0.781	1.00	23.90
Berlin	0.963	0.794	1.00	27.75
Camera	0.881	0.797	1.00	26.75
Laptop	0.898	0.729	0.75	44.85

These results show that high agreement rate, high estimation accuracy, and high score are related but not identical. Berlin had the highest score, while Laptop showed lower tau and lower agreement rate. Thus, future improvements should not only increase the overall average, but also improve worst-case behavior in difficult domains.

5 Tournament Results

In the latest tournament checked on June 23, 2026, MajiKayoAgent ranked 11th among 40 agents. Table 2 shows the top ten agents and MajiKayoAgent. All agents were evaluated over 1280 negotiations. The table includes Q1, Q3, and IQR as quartile-range statistics. IQR is defined as $Q3 - Q1$ and represents the spread of the middle 50% of the score distribution. When mean scores are close, the median and IQR help distinguish between agents that obtain moderate scores consistently and agents whose mean is lifted by a smaller number of high-score negotiations.

MajiKayoAgent produced no internal exceptions, indicating that it was stable in the tournament. However, its score was 699.24 points below the first-ranked agent and 143.88 points below the tenth-ranked agent. Its median score was 3372, which is lower than the agents above it. Since its Q3 score was 6869 and was not far from the upper group, the agent can still obtain high scores in favorable negotiations. The main weakness is that it does not sufficiently avoid low-score negotiations.

The quartile statistics give more detail. MajiKayoAgent’s IQR was 4357, close to those of BadIronV4 and AaNanteLucky. This indicates that its performance varies substantially depending on the opponent and domain. In contrast, AdaptiveBathNegotiator had an IQR of 2669 and a median of 4976, suggesting that it obtained scores more consistently without relying on a small number of very high-score negotiations. For MajiKayoAgent, improving Q1 and the median is more important than simply increasing Q3.

The average running time was 4696.21 ms, which was much larger than that of many top-ranked agents. This is mainly because the agent scans candidate outcomes and updates op-

Table 2: Main results in the latest tournament

Agent	Rank	Score	Min	Q1	Median	Q3	Max	IQR	Time[ms]
ChangAgent	1	5154.44	0	2620	5478	6924	10000	4304	406.46
AgentNexus	2	5092.32	0	2988	5285	6846	10000	3858	1770.70
Snake	3	5019.70	0	2591	5000	7075	10000	4484	318.26
Llonel	4	4921.45	0	3128	4777	6650	10000	3522	852.84
AdaptiveBathNegotiator	5	4865.29	0	3372	4976	6041	10000	2669	202.79
NegotiatorX_ANL	6	4783.53	0	3062	4614	6295	10000	3233	1249.49
AOA007	7	4730.82	0	2543	4016	6972	9991	4429	28.22
BetterCallAgentInfinity	8	4623.27	1289	2487	3961	6833	9991	4346	354.73
BadIronV4	9	4604.85	1738	2548	3629	6921	9991	4373	31.83
AaNanteLucky	10	4599.08	1384	2517	3822	6905	9994	4388	65.11
MajiKayoAgent	11	4455.20	0	2512	3372	6869	10000	4357	4696.21

ponent utility estimates during responses. Pre-sorting candidates, caching estimated utilities, and shrinking the late-stage candidate set are necessary to reduce this cost. This computational cost is not just an efficiency issue. Expensive response-time logic can become a source of instability when the execution environment changes or when the opponent implementation is also heavy. Although MajiKayoAgent had no exceptions in this tournament, the gap in running time compared with the upper ranked agents remains a clear weakness.

6 Discussion and Future Work

The results show that real-utility-based monotonic concession is effective for obtaining a stable mid-to-high tournament rank without exceptions. Nevertheless, MajiKayoAgent ranked 11th among 40 agents, and the gap to the top group remains significant. The two main issues are low-score negotiations and computational cost.

To reduce low-score negotiations, opponent utility estimation should be used gradually from the middle stage rather than only after the progress exceeds 0.5. The size of the high-self-utility candidate set should also be adjusted dynamically according to the remaining steps and the opponent’s concession speed. For example, if the opponent makes favorable offers early, the current policy is often sufficient. However, if the opponent is conservative and keeps demanding high self-utility until the late stage, MajiKayoAgent may wait too long before offering outcomes that are acceptable to the opponent. In such cases, opponent utility estimation should influence the candidate ranking earlier and more gradually.

Utility-function obfuscation should be used carefully: applying it to all candidates can reduce the agreement rate. A safer direction is to reorder only candidates in a self-utility-safe region, for example outcomes whose normalized self-utility is at least 0.7. This may reduce inferability while preserving a minimum level of self-utility. Moreover, the virtual utility function should not be used for acceptance decisions. Acceptance should always be based on real self-utility so that offer diversification and self-utility protection remain separate.

The computational issue can be addressed without changing the basic strategy. Self-utility does not change during a negotiation, so it should be computed once at initialization and reused. Opponent utility estimates also do not need to be computed for all outcomes after every model update. Since the final offer selection uses only a small short list, the model can evaluate only that list. This change would preserve the current behavior while reducing response time.

6.1 Interpretation of the Failure Cases

The tournament distribution suggests that MajiKayoAgent is not failing because of instability, but because some negotiations end with low utility or no useful agreement. This is different from a programming failure. The agent produces no

exceptions, and its upper quartile is close to several higher ranked agents. Therefore, the core negotiation logic can find high-value agreements in favorable cases. The weakness is that the same logic does not recover quickly enough when the opponent’s acceptable region is far from the agent’s initial self-utility ordering.

One likely cause is the fixed size of the late-stage candidate list. The agent always considers the top four candidates by self-utility before applying the opponent model. This is conservative and protects self-utility, but it can be too narrow when the opponent strongly prefers outcomes outside this short list. In such cases, the agent keeps offering outcomes that are good for itself but not good enough for the opponent. A dynamic list size would be more appropriate: the list could be small when the opponent appears cooperative and wider when the opponent repeatedly rejects high-self-utility offers.

Another possible cause is the fixed transition point at progress 0.5. If the opponent is highly conservative, waiting until the halfway point before using opponent-aware selection may be too late. On the other hand, using the opponent model from the beginning would make the agent vulnerable to noisy early observations. A better design is to use a smooth weighting function. For example, the offer score could be computed as a weighted sum of normalized self-utility and estimated opponent utility, with the opponent weight increasing as the deadline approaches. This would avoid a sudden behavioral change while still preserving early self-utility.

6.2 Practical Lessons

The result also gives a practical lesson for ANL agent development. A strategy that is easy to understand and stable can reach the upper middle of the tournament, but reaching the top group requires controlling the lower tail of the score distribution. In this tournament, the median and Q1 were more informative than the maximum score. MajiKayoAgent reached a maximum of 10000, but its Q1 and median were low. Future development should therefore evaluate new variants not only by average score, but also by the improvement in Q1, median, and running time.

6.3 Candidate Improvements

A direct improvement is to replace the fixed top-four late-stage list with an adaptive list. The list size can be determined from three signals: the number of remaining steps, the number of rejected self offers, and the improvement in the opponent’s offers from the agent’s point of view. If many self offers have already been rejected and the deadline is close, the agent should widen the candidate list and give the opponent model more influence. If the opponent is already making offers with high self-utility, the agent can keep the list narrow and preserve self-utility.

Another improvement is to separate exploration and exploitation. In the current agent, most offers are selected by a deterministic utility order. This makes the agent stable, but it also means that the visible offer sequence can be predictable. A small amount of controlled exploration within a high-self-utility band may help reveal which issue values the opponent accepts without losing too much utility. The exploration should be bounded by a minimum self-utility floor, because unbounded randomization was not reliable in preliminary tests.

The opponent model can also be improved by distinguishing between offers made early and offers made late. Early opponent offers may represent aspiration levels rather than realistic acceptance regions, while late offers are often closer to the opponent's true concession boundary. A time-weighted update rule could therefore give late offers more influence. Rejected self offers should also be weighted by time: a rejection near the deadline is stronger evidence than a rejection at the beginning of the negotiation.

Finally, the agent should include a lightweight risk detector. If the opponent has rejected many high-self-utility offers and the current predicted agreement probability is low, the agent should switch to a recovery mode. In recovery mode, the objective is not to maximize the next offer's self-utility, but to avoid a very low final score. Such a mode could improve Q1 and the median, which are exactly the weak points shown in the tournament table.

7 Conclusion

This paper presented the design and evaluation of MajiKayoAgent for ANL. The agent combines strong early self-utility offers, real-utility-based monotonic concession, and late-stage opponent-utility-aware offer selection. In the latest tournament, it ranked 11th among 40 agents. Future work will focus on reducing low-score negotiations and speeding up candidate evaluation.