

NegotiatorX_ANL: A Deterministic Agent for ANL 2026

Serhat Giydiren
Ozyegin University
serhat.giydiren@ozu.edu.tr
June 2026

Abstract

This report describes my final agent for the ANAC 2026 Automated Negotiation League (ANL). The submitted agent is `NegotiatorXAnl`, available as `negotiatorx_anl.agent.NegotiatorXAnl`. It is a deterministic agent: it does not use deep learning, reinforcement learning, or online model training. The final design uses a behavioral negotiation core with opponent modeling, target-based bidding, careful acceptance, and late risk control. The main goal was to get strong agreements while keeping the lower tail under control.

1 Introduction

ANL is a bilateral automated negotiation league in ANAC 2026 [1]. In each negotiation, the agent must choose offers, decide whether to accept the current offer, and provide an estimate of the opponent's utility function. The score is not only about our own agreement utility. It also depends on how well the agent handles the information available during the negotiation.

I chose a deterministic approach for practical reasons. Hidden opponents and generated domains make it risky to depend on a fragile learned model. A clear rule-based agent is easier to inspect, tune, and package. The main techniques used are rule-based control, search over outcomes, simulation, utility-based game reasoning, economic threshold logic, and offline optimization by repeated local tests.

2 Agent Design

At the beginning of a negotiation, `NegotiatorXAnl` builds a catalog of outcomes. Each outcome is normalized by our own utility, using the reserved value as the zero point. This lets the agent search over good offers without recomputing the full domain at every step. For large domains, the catalog and the exported model are bounded so the agent remains safe to run in tournaments. The opponent model is recency-weighted. If the opponent often asks for a value on an issue, that value receives more weight. The model also considers the order of offers: early repeated offers are treated as stronger signals than late concessions. It tracks repetition, concession slope, and rejected self offers. These signals are used both for bidding and for the exported `opponent_ufun`. This follows the general idea of opponent modeling in automated negotiation [2], but the implementation is small and deterministic. The bidding policy has four phases: anchor, explore, bridge, and close. Early offers protect our utility. In the middle of the negotiation, the agent still keeps a high target, but it also searches for offers that may fit the opponent. Near the end, it gives more weight to agreement risk and opponent fit. The target curve follows the standard time-dependent negotiation idea [3], but the final parameters were selected with local tournament evidence.

3 Acceptance and Risk Control

The acceptance policy compares the current offer with the planned counteroffer and with the current target. A strong offer can be accepted early. A weak offer is usually rejected unless the negotiation is late and the opponent behavior suggests that waiting is risky. This avoids accepting bad deals just because time is passing.

The agent also has a lower-tail guard. This guard becomes active only in very late repeated or hardline cases. It tries to avoid zero or very low outcomes by looking for a safe compromise that is still acceptable for our utility. This is important in ANL because a strategy with a good average score can still fail if its Q1 or no-agreement rate is poor.

For ANL submission, the agent keeps the public class path `negotiatorx_anl.agent.NegotiatorXAnl`. The submitted code also keeps a defensive response wrapper. If an unexpected error occurs during a turn, the agent falls back to a safe offer instead of letting an exception escape. The `opponent_ufun` is exported through `private_info` as a `MappingUtilityFunction`; for large domains the mapping is capped and focused on high-utility and recently observed outcomes.

4 Evaluation

I evaluated the final candidate in three steps. First, I ran unit and command line tests to check packaging and basic negotiation behavior. Second, I tested official fixed scenarios one by one against the provided local opponents. Third, I used generated scenarios to check whether the agent still works on unseen domains. The local opponents are not the full competition field, but they are a useful regression guard.

The command-line and import checks passed. The focused test suite `tests/test_mynegotiator.py tests/test_cli.py` passed with 25 tests. The final `submission.zip` contains only `negotiatorx_anl/` and `requirements.txt`. Importing `negotiatorx_anl.agent.NegotiatorXAnl` from the zip also passed.

One all-scenario parallel local tournament did not finish cleanly. It stopped making progress after 120 of 224 negotiation files were written. I treated this as a local parallel-runner issue because all fixed scenarios finished when they were run separately.

Check	Result	Score
Amsterdam fixed tournament	rank 1	1.3116
Camera fixed tournament	rank 1	1.3117
Laptop fixed tournament	rank 1	1.4144
Car fixed tournament	rank 2	1.3981
Grocery fixed tournament	rank 3	1.3402
ISBTAcquisition fixed tournament	rank 3	1.2974
NiceOrDie fixed tournament	rank 4	0.5164
Generated 3-scenario tournament	rank 1	1.1673

Table 1: Final local tournament checks with the submitted agent.

5 Conclusion

`NegotiatorXAnl` is a deterministic behavioral negotiation agent for ANL 2026. Its main strengths are stable bidding, an explicit opponent model, careful acceptance, and a late lower-tail guard. The design does not depend on external data files or heavy online learning. The final local checks show that the agent is package-safe and competitive on generated scenarios, while still having weak spots on some fixed domains. The clearest future improvement would be a better policy for very harsh domains such as NiceOrDie, without increasing the no-agreement rate.

References

- [1] ANAC 2026 Automated Negotiation Competition. <https://anac.cs.brown.edu/>, 2026. Competition documentation.
- [2] Tim Baarslag, Mark J. C. Hendriks, Koen V. Hindriks, and Catholijn M. Jonker. Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, 30:849–898, 2016.
- [3] Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3):159–182, 1998.