

ErmanConcealNegotiator: A Preference-Concealing Agent for ANL 2026

David Erman

June 2026

Abstract

This report describes **ErmanConcealNegotiator**, an autonomous agent for the ANL 2026 bilateral negotiation league. The agent is designed for the alternating-offers protocol and for the 2026 objective: reaching high-value agreements while making the observed sequence of bids less revealing about the agent’s exact private preferences. The agent uses a Boulware-style concession curve, a per-negotiation frequency model of the opponent’s offers, an acceptance rule based on the utility of the next planned offer, and a deliberate bid-diversification rule that avoids repeatedly proposing the same maximum-utility outcome. The implementation has no external dependencies, performs no file I/O, and stores no information across negotiations.

1 Competition setting and design goals

ANL 2026 evaluates agents in bilateral multi-issue negotiations under the alternating-offers protocol. In each round, an agent can accept the current offer, reject it and make a counteroffer, or end the negotiation. The score depends on the obtained agreement and on the ability to handle preference uncertainty. Therefore, the agent must balance two goals. First, it should obtain an agreement with high utility above the reservation value. Second, it should not reveal its utility function too directly through a predictable sequence of best offers.

The design goals are:

- **Robust validity:** the agent should be simple, deterministic enough to debug, and independent of extra packages.
- **High utility:** the agent should demand very good outcomes early and concede mainly close to the deadline.
- **Agreement safety:** the agent should avoid unnecessary timeouts when the opponent offers a rational outcome late in the negotiation.
- **Preference concealment:** the agent should vary offers among a band of similarly good outcomes instead of always proposing the single best outcome.
- **Opponent awareness:** the agent should learn, inside each negotiation only, which issue values appear often in the opponent’s bids.

2 Concealing bidding strategy

Let $t \in [0, 1]$ be the relative negotiation time and let r be the reservation value. The agent computes a target utility level

$$A(t) = \max\{r + 0.015, e + (s - e)(1 - t^{3.2})\},$$

where s is a high starting target close to the best available utility and $e = \max\{r + 0.055, 0.58\}$ is the final planned target. This is a Boulware-style curve: the target stays high for most of the negotiation and decreases more quickly near the end.

At each proposing turn, the agent considers outcomes whose own utility is at least $A(t)$. If this set is too small, it relaxes the target slightly while staying above the reservation value. The chosen offer maximizes a weighted score of the form

$$S(o, t) = w_1(t)U_{self}(o) + w_2(t)\hat{U}_{opp}(o) + w_3(t)D(o) - P_{rep}(o) - P_{rev}(o),$$

where U_{self} is normalized self-utility, $\hat{U}_{opp}$ is the estimated opponent utility, D rewards diversity from recent own offers, P_{rep} penalizes repeating the same bid, and P_{rev} penalizes offering only extremely high self-utility outcomes when several less revealing high-quality alternatives are available.

This bidding rule tries to conceal preferences in a practical way. If an agent always sends its best outcome, an opponent can infer important issue weights quickly. Instead, this agent sends a rotating set of offers that are still good for itself but differ in issue values. This creates a noisier signal while still leaving enough utility for a strong agreement.

3 Opponent modeling

The opponent model is a per-negotiation frequency model. Whenever the opponent proposes an offer, the agent records the issue values appearing in that offer. Later offers receive slightly larger weight because they are more likely to describe what the opponent is willing to accept near the deadline. For an outcome $o = (o_1, \dots, o_m)$, the estimated opponent utility is the average smoothed frequency score of the values o_i :

$$\hat{U}_{opp}(o) = \frac{1}{m} \sum_{i=1}^m \frac{c_i(o_i) + 1}{T_i + d_i},$$

where $c_i(v)$ is the weighted count of value v at issue i , T_i is the total weighted count for issue i , and d_i is the number of distinct values observed for issue i . The smoothing term prevents unseen values from receiving zero score.

The model is deliberately simple. It uses only information from the current negotiation and does not save memory between different negotiations. It is therefore consistent with the competition rule forbidding persistent learning across negotiations.

4 Acceptance strategy

The acceptance rule compares the current opponent offer with three benchmarks:

1. the current aspiration level $A(t)$;
2. the utility of the next offer that the agent would make itself;
3. a deadline-dependent rationality threshold above the reservation value.

An offer is accepted immediately if it is extremely good. Otherwise, the agent accepts if the offer is at least as good as the next planned counteroffer, or if it passes the current aspiration threshold. Near the deadline, the agent becomes more willing to accept rational offers above the reservation value. This prevents the common failure mode of losing a good agreement because the agent waited for a slightly better counteroffer too long.

5 Implementation details

The implementation is contained in one file, `erman_conceal_negotiator.py`. The class name is `ErmanConcealNegotiator`; the module name intentionally avoids the forbidden placeholder names `mynegotiator` and `MyNegotiator`. The code subclasses the standard `NegMAS SAONegotiator` and implements the usual `propose` and `respond` callbacks. It also exposes a callable opponent-model proxy through `opponent_model`, together with methods `estimate_opponent_utility` and `opponent_utility`.

For efficiency, the agent caches the outcome space at the beginning of a negotiation when enumeration is available. If enumeration fails or is too large, it uses bounded random sampling through the negotiation mechanism interface. The maximum cached set is limited to keep runtime safe.

6 Expected strengths and limitations

The expected strengths are robustness, low runtime cost, strong early bargaining, and a less revealing offer sequence. The agent should be safer than a pure hardliner because it concedes late and accepts good rational offers. It should also be less transparent than a pure time-based negotiator because it rotates among high-quality outcomes and uses opponent-frequency information when selecting offers.

The main limitation is that the opponent model is lightweight. It can be effective against opponents whose bids reveal stable issue-value preferences, but it is not a full probabilistic reconstruction of the opponent’s utility function. A stronger version would run offline parameter tuning over the official scenarios and compare several concession curves and acceptance thresholds. Under the short available time, the current version prioritizes a valid and competitive submission over a complex model that might fail the portal tests.

7 Submission information

Recommended portal fields:

- Agent name: `Erman Conceal Negotiator`
- Agent alias: `ErmanConceal`
- Agent module: `erman_conceal_negotiator`
- Agent class: `ErmanConcealNegotiator`
- Dependencies: none

The agent should be tested with `anl2026 run` and `anl2026 tournament` before final upload.