

EmEfAgent: Preference-Concealing Negotiation for ANL 2026

AgentInOzu Team (Group 361)

Department of Computer Science
Ozyegin University

CS 451: Automated Negotiation — Spring 2026

Abstract

Automated bilateral negotiation requires agents to balance two competing objectives: maximising own utility while concealing preferences from the opponent. We present EmEfAgent, a rule-based negotiation agent designed for the ANL 2026 competition that addresses this tension explicitly. The agent follows a BOA (Bidding, Opponent modelling, Acceptance) architecture implemented as a subclass of `SAONegotiator` in the `NegMAS` library. Its central contribution is a utility-preserving deceptive bidding strategy that targets the agent’s highest-weight issue to maximally mislead frequency-based opponent models without sacrificing agreement utility. This is complemented by an entropy-weighted frequency opponent model, adaptive Boulware concession, and a multi-rule acceptance strategy. EmEfAgent requires no training data or external services, making it a fully explainable and reproducible competition submission.

Keywords: automated negotiation, preference concealment, Boulware strategy, opponent modelling, ANL 2026

1 Introduction

Automated bilateral negotiation requires agents to balance two competing objectives. On one hand, an agent should maximise its utility by proposing and accepting outcomes that are as close as possible to its own ideal point. On the other hand, the ANL 2026 scoring scheme also rewards agents for *concealing* their preferences from the opponent: a well-concealing agent makes it harder for the opponent’s model to infer its true utility function, thereby reducing the opponent’s ability to exploit that information in later rounds or in future sessions.

This tension creates a fundamental trade-off. Repeatedly offering the outcomes that maximise own utility reveals the agent’s preference ordering; deceptive offers may obscure that ordering but can lower agreement quality or risk no-deal outcomes. EmEfAgent is designed to navigate this trade-off explicitly.

The agent follows a BOA (Bidding, Opponent modelling, Acceptance) architecture [1] and is implemented in Python as a subclass of `SAONegotiator` from the `NegMAS` library [4]. It is entirely rule-based: no large language models, reinforcement learning, or external services are used.

The remainder of this report is structured as follows. Section 2 reviews the relevant background. Section 3 gives an overview of the agent’s architecture. Sections 4–6 describe each component in detail. Section 7 covers implementation specifics. Section 8 discusses strengths and limitations, and Section 9 concludes.

2 Background

Bilateral negotiation agents typically decompose their decision-making into three cooperating components: a bidding strategy that determines what to offer, an opponent model that infers the other party’s preferences, and an acceptance strategy that decides when to agree [1].

Time-dependent concession functions, particularly the Boulware family introduced by Faratin et al. [3], are a standard tool for bidding strategy design. Frequency-based opponent modelling—inferring value preferences from offer frequencies—remains one of the most widely deployed approaches due to its simplicity and zero prior knowledge requirements.

The ANL 2026 competition [4] adds a concealment objective to the classical utility-maximisation goal, rewarding agents whose offers prevent opponents from accurately modelling their true preferences. This dual objective motivates the design of EmEfAgent.

3 Architecture Overview

EmEfAgent is structured around four cooperating components, summarised in Table 1.

At each negotiation step the agent: (1) receives the opponent’s offer and updates the opponent model (`respond`); (2) evaluates the offer against the acceptance rules and either accepts or rejects; (3) if rejecting, generates its own counter-offer (`propose`), choosing between an honest bid and a deceptive bid.

Table 1: Main components of EmEfAgent.

Component	Method / Strategy
Bidding	Boulware target with utility-preserving deceptive bids
Opponent Modelling	Frequency model with entropy-based issue weights
Acceptance	Reservation floor, ACNext, Nash-product, deadline panic
Concealment	Misleading values on highest-weight issue
Learning type	Rule-based; no LLM or external training

4 Bidding Strategy

4.1 Boulware Target Utility

EmEfAgent uses a time-dependent target utility inspired by the Boulware family of concession strategies [3]. Let $t \in [0, 1]$ denote the normalised negotiation time ($t = \text{step}/n_steps$). The target utility is:

$$u_{\text{target}}(t) = u_{\min} + (u_{\max} - u_{\min}) \left(1 - t^{1/e}\right), \quad (1)$$

where e is a concession-speed exponent (default $e = 0.20$), $u_{\max} = 1.0$, and

$$u_{\min} = \max(\text{min_utility}, u_{\text{reserved}} + 0.15). \quad (2)$$

A small value of e produces a very Boulware profile: the agent stays close to $u_{\max}$ for most of the negotiation and concedes sharply only near the deadline.

4.2 Behaviour-Adaptive Concession Rate

Once at least 15 opponent bids have been observed, the concession exponent is micro-adjusted based on the estimated opponent style:

- **Very Boulware opponent** ($\hat{e}_{\text{opp}} < 0.15$): $e \leftarrow \min(1.10 e_{\text{base}}, e_{\text{base}} + 0.02)$. A slight increase prevents mutual deadlock.
- **Fast conceder** ($\hat{e}_{\text{opp}} > 3.0$): $e \leftarrow \max(0.95 e_{\text{base}}, e_{\text{base}} - 0.01)$. The opponent is moving towards us; we can afford to wait.
- **Otherwise**: e reverts to e_{base} .

Adjustments are capped at $\pm 10\%$ of e_{base} to avoid overfitting to noisy early observations.

4.3 Target Adaptation to Opponent Behaviour

The target computed by Equation 1 is further adapted based on a coarser behaviour classification (Section 5):

- **Conceder**: target is raised by 0.05 (capped at 0.95) to exploit the opponent’s willingness to concede.
- **Boulware** (and $t > 0.70$): target is lowered by 0.04 to avoid no-deal when both parties are entrenched late in the session.

4.4 Honest and Deceptive Bidding

All discrete outcomes are pre-sorted by own utility at negotiation start. An *honest bid* is the highest-utility outcome at or above the current (adapted) target. If no such outcome exists, the agent falls back to the best outcome above the reservation value.

With probability $p_{\text{dec}} = 0.25$ and only within the time window $t \in (0.10, 0.88)$, the agent substitutes a *deceptive bid* for the honest bid. The deceptive bid must still satisfy the current target (own utility is not sacrificed) but is chosen to maximally confuse frequency-based opponent models:

1. Identify the issue i^* with the highest weight in the agent’s own utility function, $i^* = \arg \max_i w_i$.
2. Among all outcomes above the target, select the one that uses the *worst* value for the agent on issue i^* .

The intuition is that frequency-based opponent models infer preferences from offer frequencies. By repeatedly showing the worst value on the most important issue while maintaining acceptable utility, the agent creates a concentrated mis-signal on that issue, causing the opponent’s model to rank the agent’s most-preferred outcomes as least preferred. The deceptive bid is skipped if no acceptable outcome can be found; the agent then falls back to an honest bid, so utility is never sacrificed for concealment.

5 Opponent Modelling

5.1 Frequency-Based Value Weights

EmEfAgent maintains a `FreqModel` object that tracks the number of times each value v of each issue i has appeared in the opponent’s offers. After every update the normalised frequency serves as the value weight:

$$\hat{w}_i(v) = \frac{c_i(v)}{\sum_{v'} c_i(v')}, \quad (3)$$

where $c_i(v)$ is the count of value v on issue i .

5.2 Entropy-Based Issue Weights

Issue importance is inferred from the concentration of the opponent’s value distribution. For issue i the Shannon entropy of its value-weight distribution is:

$$H_i = - \sum_v \hat{w}_i(v) \log(\hat{w}_i(v) + \varepsilon), \quad (4)$$

with $\varepsilon = 10^{-12}$ for numerical stability. Low entropy signals that the opponent strongly concentrates on a few values, suggesting that issue is important to them. The issue weight is therefore:

$$W_i = \frac{H_{\max} - H_i}{\sum_j (H_{\max} - H_j)}, \quad H_{\max} = \max_j H_j. \quad (5)$$

The estimated utility of an outcome o is then:

$$\hat{u}_{\text{opp}}(o) = \sum_i W_i \hat{w}_i(o_i). \quad (6)$$

5.3 Behaviour Classification

After at least 5 bids have been received, the model classifies opponent behaviour from the time series of estimated utilities $\{\hat{u}_{\text{opp}}(o_k)\}$:

- **Conceder**: mean difference < -0.015 .
- **Boulware**: $|\text{mean difference}| < 0.008$ and $\text{std} < 0.02$.
- **Random**: $\text{std} > 0.05$.
- **Selfish**: otherwise.

5.4 Concession Exponent Estimation

The model estimates the opponent's Boulware concession exponent $\hat{e}_{\text{opp}}$ using the Faratin model [3]:

$$u(t) = u_{\min} + \Delta(1 - t^{1/\hat{e}}), \quad (7)$$

where $\Delta = u_{\max} - u_{\min}$. Rearranging,

$$\hat{e} = \frac{\ln t}{\ln(1 - r)}, \quad r = \frac{u - u_{\min}}{\Delta}, \quad (8)$$

and the final estimate is the median over all valid (t, u) pairs with $t \in (0.05, 0.95)$ and $r \in (0.05, 0.95)$.

6 Acceptance Strategy

The acceptance function evaluates four sequential rules.

Rule 0 — Reservation floor. Any offer below the reservation value is immediately rejected: $u_{\text{offer}} < u_{\text{reserved}} \Rightarrow \text{REJECT}$.

Rule 1 — ACNext. If the offer is at least as good as the outcome the agent would currently propose (i.e. $u_{\text{offer}} \geq u_{\text{target}}$), acceptance is rational [2]:

$$u_{\text{offer}} \geq u_{\text{target}}(t) \Rightarrow \text{ACCEPT}. \quad (9)$$

Rule 2 — Nash-point acceptance. After $t > 0.70$ and at least 10 opponent bids have been observed, the agent periodically estimates the Nash bargaining product:

$$N^* = \max_{o: u_{\text{self}}(o) \geq u_{\text{reserved}}} u_{\text{self}}(o) \cdot \hat{u}_{\text{opp}}(o), \quad (10)$$

computed over the top-300 outcomes by own utility. An offer is accepted if

$$u_{\text{offer}} \cdot \hat{u}_{\text{opp}}(\text{offer}) \geq (1 - r_{\text{Nash}}) N^*, \quad r_{\text{Nash}} = 0.05, \quad (11)$$

provided the offer is not more than 10% below the current ACNext target.

Rule 3 — Deadline panic. When $t > 0.90$, a relaxed floor $u_{\text{panic}} = \max(u_{\text{reserved}} + 0.05, 0.85 u_{\min})$ is used, accepting any offer above it. This reduces no-deal risk while still rejecting clearly poor offers.

7 Implementation

7.1 Class Structure

The agent is implemented as two Python classes:

- **FreqModel**: standalone opponent model callable as a utility function (`__call__`), satisfying the ANL 2026 interface requirement for `opponent_ufun`.
- **EmEfAgent (SAONegotiator)**: main agent class following the NegMAS SAO interface.

7.2 Initialisation

`on_negotiation_start` performs the following setup: (1) instantiates `FreqModel` with the negotiation issues; (2) computes `_effective_min = max(min_utility,  $u_{\text{reserved}} + 0.15$ )`; (3) pre-sorts all discrete outcomes by own utility (descending), enabling $O(n)$ bid selection; and (4) identifies the highest-weight issue i^* for deceptive bidding.

7.3 Hyperparameters

Table 2: Default hyperparameter values.

Parameter	Default	Description
e	0.20	Boulware concession exponent
p_{dec}	0.25	Deception probability
<code>min_utility</code>	0.52	Soft utility floor
r_{Nash}	0.05	Nash-acceptance radius

7.4 Computational Complexity

At each step, the opponent model update is $O(n_{\text{issues}})$ and bid selection is $O(n_{\text{outcomes}})$ in the worst case, but in practice the pre-sorted list allows early termination. The Nash product refresh scans the top-300 outcomes every 5 bids, giving an amortised $O(300/5) = O(60)$ overhead per step.

7.5 Key Code Excerpt

The core deceptive bidding logic is shown in Listing 1.

Listing 1: Deceptive bid selection (simplified).

```

1 def _deception_bid(self, min_utility:
2     float) -> Optional[Outcome]:
3     acceptable = [o for o in self.
4         _sorted_outcomes
5         if float(self.ufun(o))
6             >= min_utility]
7     my_weights = self._my_weights()
8     target_issue = max(range(len(
9         my_weights)),
10         key=lambda i:
11             my_weights[i])
12     all_vals = list(iter(self.nmi.issues
13         [target_issue]))
14     worst_val = sorted(all_vals,
15         key=lambda v: float(
16             self.ufun.values[
17                 target_issue](v)))[0]
18     for o in acceptable[:300]:
19         if o[target_issue] == worst_val:
20             return o
21     return None # fallback to honest
22     bid

```

8 Discussion

8.1 Strengths

Utility-preserving deception. Unlike approaches that sacrifice utility to mislead opponents, EmEfAgent’s deceptive bids always satisfy the current target. Concealment is a free by-product of bid selection, not a cost.

Focused mis-signalling. Targeting a single high-weight issue concentrates the misleading frequency signal where it causes maximum damage to frequency-based models. The deception is deterministic once the target value is identified, ensuring reproducibility.

Robustness to model inaccuracy. The Nash-point acceptance rule is gated behind time ($t > 0.70$) and bid-count (≥ 10) thresholds, so the agent does not rely on a potentially unreliable early opponent model for critical decisions.

Simplicity and explainability. The agent has no learned parameters and no external dependencies beyond NegMAS. Every decision rule can be inspected directly in the source code.

8.2 Limitations

Frequency-model assumption. The deceptive strategy is effective only against opponents that use frequency-based preference inference. Against opponents using different modelling approaches (e.g. Bayesian inference over utility functions, or no modelling at all), the concealment benefit may be reduced.

Noisy opponent model. Against highly random or adversarial opponents, the frequency model may not converge to a useful estimate. The entropy-based issue weighting partially mitigates this by down-weighting uninformative issues, but early-stage inaccuracy remains a concern.

Fixed deception window. The time window $t \in (0.10, 0.88)$ and probability $p_{\text{dec}} = 0.25$ are fixed hyperparameters. In future work, these could be adapted based on observed opponent modelling accuracy or on the magnitude of the ANL 2026 concealment reward.

Discrete outcome space. The current implementation enumerates all discrete outcomes at start-up. For domains with very large or continuous outcome spaces this is infeasible; a sampling-based fallback is available in the deception code but has not been optimised.

9 Conclusion

EmEfAgent is a rule-based automated negotiation agent designed for the ANL 2026 competition. Its central contribution is a utility-preserving deceptive bidding strategy that targets the agent’s most important issue to maximally mislead frequency-based opponent models, without sacrificing agreement utility. This is complemented by an entropy-weighted frequency opponent model, adaptive Boulware concession, and a multi-rule acceptance strategy that progresses from conservative to panic-mode as the deadline approaches.

The agent is simple, fully explainable, and requires no training data or external services, making it a robust and reproducible submission for the ANL 2026 tournament.

References

- [1] Baarslag, T.; Hendriks, M.; Hindriks, K.; and Jonker, C. 2016. Learning about the Opponent in Automated Bilateral Negotiation: A Comprehensive Survey of Opponent Modeling Techniques. *Autonomous Agents and Multi-Agent Systems*, 30(5): 849–898.
- [2] Baarslag, T.; Hindriks, K.; and Jonker, C. 2011. Acceptance Conditions in Automated Negotiation. In *Proceedings of the AAMAS Workshop on Agent-based Complex Automated Negotiations*.
- [3] Faratin, P.; Sierra, C.; and Jennings, N. R. 1998. Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, 24(3–4): 159–182.
- [4] Mohammad, Y. 2020. NegMAS: A Platform for Automated Negotiations. <https://github.com/yasserfarouk/negmas>. Accessed: 2026-05-01.