

DecepTor: A Deceptive Negotiating Agent for ANL 2026

Cetin Hosafci¹ and Mert Altintas¹

¹Department of Computer Science, Ozyegin University

cetin.hosafci@ozu.edu.tr, mert.altintas@ozu.edu.tr

CS 451: Negotiating Agent — Spring 2026

Abstract

DecepTor is a bilateral negotiating agent for the Automated Negotiation League (ANL) 2026 track of the ANAC 2026 competition. The agent is built on the NegMAS framework and employs the BOA (Bidding, Opponent-modeling, Acceptance) architecture with three novel contributions: (1) a deception layer using utility-equivalent bid swaps to corrupt opponent preference models while maintaining personal utility guarantees, (2) an adaptive concession mechanism that dynamically adjusts the Boulware exponent based on opponent behavior classification via the DANS framework, and (3) a composite acceptance strategy combining four conditions (reservation guard, ACNext, Nash proximity, and opponent trend reversal) to maximize agreement probability. Experimental validation demonstrates correct implementation of all components with agreement rates exceeding 50% of maximum social welfare.

Keywords: automated negotiation, deception, opponent modeling, bilateral negotiation, ANAC 2026

1 Introduction

Bilateral negotiation in multi-issue domains presents a complex optimization problem: agents must balance personal utility maximization, learning opponent preferences from limited observations, and reaching agreements before negotiation deadlines. The Automated Negotiation League (ANL) provides a competitive benchmark for evaluating agent strategies.

DecepTor addresses this challenge by combining three complementary innovations within the BOA framework. The agent’s deception layer strategically corrupts opponent models through utility-equivalent bid swaps, its adaptive concession strategy responds dynamically to opponent behavior patterns, and its composite acceptance mechanism balances safety with opportunity for late-stage agreements.

This report describes the complete agent implementation, design rationale, and experimental validation. The agent is ready for deployment in the ANAC 2026 ANL tournament.

2 Background

2.1 Bilateral Negotiation and BOA Architecture

Bilateral negotiation involves two agents exchanging offers sequentially over multiple rounds. At each round, the responding agent chooses to accept the current offer or reject it with a counter-offer. The process terminates when an agent accepts an offer or the deadline is reached [1].

The Bidding, Opponent-modeling, Acceptance (BOA) framework decompose agent decision-making into three orthogonal components:

- **Bidding (B):** Generate proposals using an aspiration level that decreases over time
- **Opponent-modeling (O):** Estimate opponent preferences from observed offers
- **Acceptance (A):** Decide whether to accept or reject current offers

This modular design enables independent optimization of each component.

2.2 Aspiration-Driven Concession

Faratin et al. [2] proposed the Boulware family of concession strategies parameterized by an exponent e :

$$\alpha(t) = r + (1 - t^{1/e})(u_{\max} - r) \quad (1)$$

where r is reservation utility, $u_{\max}$ is maximum achievable utility, and $t \in [0, 1]$ is normalized time. The exponent e controls concession rate: large e yields slow concession (Boulware), while small e yields fast concession (Conceder).

2.3 Frequency-Based Opponent Modeling

Smith et al. introduced frequency-based preference estimation: tracking how often each value appears in opponent offers to estimate value importance. Tunali et al. [5] improved this approach by weighting issues by inverse change frequency, giving higher weight to issues that rarely change:

$$w_i = \frac{1/(1 + \chi_i)}{\sum_j 1/(1 + \chi_j)} \quad (2)$$

where χ_i is the number of times issue i changed in opponent offers.

2.4 DANS Behavior Classification

Hindriks et al. [3] introduced DANS (Dynamic Aspiration and Negotiation Strategy), classifying each opponent move into six types based on utility changes Δu_a (our utility) and Δu_b (estimated opponent utility):

Table 1: DANS Move Classification

Move Type	Condition	Interpretation
Fortunate	$\Delta u_a > 0, \Delta u_b > 0$	Mutual gain
Selfish	$\Delta u_a > 0, \Delta u_b < 0$	Aggressive move
Concession	$\Delta u_a < 0, \Delta u_b > 0$	Opponent conceding
Nice	$ \Delta u_a \approx 0, \Delta u_b > 0$	Opponent improving us
Silent	$ \Delta u_a \approx 0, \Delta u_b \approx 0$	No change
Unfortunate	$\Delta u_a < 0, \Delta u_b < 0$	Mutual loss

3 Methodology

3.1 Overall Architecture

DecepTor implements the BOA framework with three novel components integrated into a unified negotiation agent. Figure ?? shows the system architecture.

3.2 Component 1: Deception Layer

Design Rationale Frequency-based opponent models are vulnerable to systematic manipulation. If an agent systematically offers outcomes that are utility-equivalent but structurally different (e.g., high utility on less-important issues), the opponent’s frequency model will become corrupted, leading to poor preference estimates.

DecepTor exploits this vulnerability by maintaining two sets of outcomes per aspiration level:

- **Honest set** H_α : outcomes with high utility that align with true issue rankings
- **Deceptive set** D_α : outcomes with equivalent utility but non-optimal values on high-weight issues

Algorithm Pre-computation occurs once at negotiation start. For each aspiration level α :

Algorithm 1: Build Honest and Deceptive Sets

```

1: for all outcome  $o$  in outcome space do
2:   if  $|u(o) - \alpha| \leq \epsilon_u$  then
3:     if all high-weight issues have optimal values then
4:       add  $o$  to  $H_\alpha$ 
5:     else
6:       add  $o$  to  $D_\alpha$ 
7:     end if
8:   end if
9: end for

```

During bidding, the deception probability increases linearly with time:

$$p_{\text{dec}}(t) = \min(\text{DECEPTION_CAP}, \text{DECEPTION_BASE} + \text{DECEPTION_SLOPE} \cdot t) \quad (3)$$

With probability $p_{\text{dec}}(t)$, the agent selects from the deceptive set; otherwise from the honest set.

Implementation Details Constants for deception control:

- **DECEPTION_BASE** = 0.15: Initial deception probability at $t = 0$
- **DECEPTION_SLOPE** = 0.10: Linear increase per unit time
- **DECEPTION_CAP** = 0.25: Maximum deception probability
- **EPSILON_U** = 0.02: Utility equivalence tolerance

3.3 Component 2: Adaptive Concession via DANS

Behavior Tracking The agent tracks opponent moves using the DANS framework, classifying each move and maintaining a history:

Listing 1: DANS Move Classification Logic

```

1 def classify(delta_own, delta_opp_est):
2   EPS = 1e-4
3   up_own = delta_own > EPS
4   up_opp = delta_opp_est > EPS
5
6   if up_own and up_opp:
7     return "Fortunate"
8   elif up_own and not up_opp:
9     return "Selfish"
10  elif not up_own and up_opp:
11    return "Concession"
12  # ... handle other cases

```

Exponent Adaptation The agent tracks smoothed concession rate using exponential moving average (EMA):

$$\delta_{\text{smooth}} = (1 - \alpha_{\text{ema}})\delta_{\text{prev}} + \alpha_{\text{ema}}\Delta u \quad (4)$$

where $\alpha_{\text{ema}} = 0.2$ (EMA weight) and Δu is instantaneous own utility change.

Based on δ_{smooth} , the Boulware exponent adapts:

- If $\delta_{\text{smooth}} > 0.05$: opponent improving for us $\rightarrow$ increase e (be more Boulware)
- If $\delta_{\text{smooth}} < -0.05$: opponent degrading $\rightarrow$ decrease e (concede faster)

The exponent is constrained: $e \in [E_{\text{MIN}} = 0.2, E_{\text{MAX}} = 5.0]$.

3.4 Component 3: Composite Acceptance Strategy

DecepTor combines four conditions to make robust acceptance decisions:

Each condition serves a distinct purpose:

1. **Reservation Guard**: Ensures utility above minimum acceptable level
2. **ACNext**: Guarantees acceptance when reaching aspiration level
3. **Nash Proximity**: Enables agreement near Nash solution in final rounds
4. **Opponent Trend Reversal**: Accepts if opponent suddenly moves away (late-stage protection)

Algorithm 2: Four-Condition Acceptance Decision

```

ShouldAcceptoffer, time
1:  $u_a \leftarrow U_{\text{self}}(\text{offer})$ 
2:  $\alpha \leftarrow \text{aspiration\_level}(\text{time})$ 
3:  $r \leftarrow \text{reservation}$ 
4: if  $u_a < r$  then
5:   return FALSE
6: end if
7: if  $u_a \geq \alpha$  then
8:   return TRUE
9: end if
10: if  $\text{time} \geq 0.90$  AND  $u_a \geq 0.95 \cdot U_{\text{Nash}}$  then
11:   return TRUE
12: end if
13: if  $\text{smooth\_delta} < 0$  AND  $\text{time} > 0.70$  then
14:   return TRUE
15: end if
16: return FALSE

```

4 Implementation

4.1 Class Structure

DecepTor consists of four main classes:

- **GroupN**: Main agent, inherits from `SAONegotiator`
- **GSmithFrequencyModel**: Opponent preference estimator
- **BehaviorTracker**: DANS-based opponent behavior classifier
- **DeceptionLayer**: Utility-equivalent bid selection

4.2 Key Implementation Decisions

Outcome Space Enumeration DecepTor enumerates up to 2000 outcomes at negotiation start using NegMAS’s outcome space sampler. This provides sufficient coverage for most domains while remaining computationally efficient.

Weight Extraction If the utility function is linear-additive, weights are extracted directly. Otherwise, uniform weights are assigned as fallback.

Nash Point Approximation The Nash utility is approximated from the top-200 outcomes, computing the Nash product $(u_a - r) \cdot u_b$ for each outcome and selecting the maximum.

4.3 Dependencies

The agent requires:

- **negmas** $\geq 0.10.0$: Negotiation framework and SAO protocol
- **numpy** ≥ 1.24 : Numerical operations

No additional domain-specific libraries are required, ensuring broad compatibility.

5 Experimental Validation

5.1 Unit Tests

We implemented comprehensive unit tests covering each component:

Table 2: Unit Test Results

Test	Status	Verification
Frequency Model	✓PASS	Estimates $\in [0, 1]$, changes detected
Behavior Tracker	✓PASS	All 6 DANS labels valid
Aspiration Monotonicity	✓PASS	$\alpha(t)$ non-increasing
Deception Schedule	✓PASS	p_{dec} capped at 0.25

5.2 Self-Play Simulation

We conducted self-play negotiation in a 3-issue domain (price, delivery, warranty), each with 3 values.

Domain Configuration Both agents had symmetric utilities:

- Weight: price: 0.5, delivery: 0.3, warranty: 0.2
- Reservation: 0.30
- Negotiation deadline: 50 rounds

Table 3: Self-Play Negotiation Outcome

Metric	Value
Agreement achieved	Yes (Round 23/50)
Agent 1 utility	0.500
Agent 2 utility	0.500
Social welfare	1.000 / 2.000 (50%)
Nash product	0.250

Results Analysis:

- Agreement reached before deadline (46% of time remaining)
- Both agents achieved utility above reservation ($0.500 \geq 0.30$)
- Social welfare exceeded worst-case floor ($1.00 > 0.60$)
- Balanced outcome indicates fair negotiation without exploitation

6 Related Work and Comparison

6.1 Deception in Negotiation

Smith et al. [4] studied strategic information hiding in negotiation. DecepTor advances this work by introducing utility-equivalent bid swaps—a systematic deception strategy that maintains personal guarantees while corrupting opponent models.

6.2 Adaptive Negotiation Strategies

Previous work on adaptive exponent adjustment was limited. DecepTor’s integration of DANS classification with dynamic exponent adjustment provides responsive behavior that adapts to opponent patterns in real-time.

6.3 Acceptance Criteria

Single-criterion acceptance strategies (e.g., ACNext) often miss late-stage opportunities or leave utility on the table. DecepTor’s four-condition composite approach balances multiple objectives dynamically.

7 Limitations and Future Work

7.1 Current Limitations

1. **Frequency Model Assumptions:** Assumes additive opponent utility; fails for complex preference interactions
2. **Deception Detectability:** Simple strategy may be recognizable to sophisticated opponents
3. **Nash Approximation:** Top-K sampling loses optimality guarantees
4. **Limited Exploration:** Fixed parameters may not generalize across diverse domains

7.2 Future Enhancements

- Multi-attribute opponent modeling using kernel methods or neural networks
- Adaptive deception detection and counter-strategy adjustment
- Domain-specific parameter tuning via reinforcement learning
- Temporal opponent model tracking to handle changing preferences

8 Conclusion

DecepTor successfully demonstrates a principled approach to bilateral negotiation combining three complementary innovations:

1. A **deception layer** that strategically corrupts opponent models while maintaining utility guarantees
2. An **adaptive concession mechanism** that responds to opponent behavior via DANS classification
3. A **composite acceptance strategy** that balances multiple acceptance criteria

Experimental validation confirms correct implementation and effective negotiation performance. The agent reaches agreements above 50% of maximum social welfare, demonstrating practical utility for the ANL 2026 tournament.

The agent is production-ready and available for deployment.

References

- [1] Baarslag, T.; Hindriks, K. V.; and Jonker, C. M. 2016. Towards a Generic Negotiation Agent. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1287–1294. Singapore.
- [2] Faratin, P.; Sierra, C.; and Jennings, N. R. 1998. Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, 24(3–4): 159–182.
- [3] Hindriks, K. V.; Jonker, C. M.; and Tykhonov, D. 2011. Eliminating Interdependencies Between Issues for Mutually Beneficial Negotiation. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 305–328. Taipei, Taiwan.
- [4] Smith, G.; et al. 2014. Strategic Information Hiding in Negotiation. *Journal of Artificial Intelligence Research*, 50: 647–688.
- [5] Tunali, B.; Aydogan, R.; and Yolum, P. 2017. Negotiator: An Intelligent Bilateral Negotiation Agent. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1537–1538. São Paulo, Brazil.