

Ianos: An Adaptive, Concealing, and Time-Aware Autonomous Negotiator for ANL 2026

Abstract

This report presents the architectural design of **Ianos**, an autonomous negotiating agent developed for the Automated Negotiation League (ANL 2026) under the Stacked Alternating Offers (SAO) protocol. Building upon its predecessor, Ianos introduces a robust, multi-layered framework designed to maximize self-advantage while systemically neutralizing opponent exploitation. The agent integrates an enhanced rank-inversion preference concealment layer, a robust outlier-filtered recency-weighted opponent frequency model, an adaptive multi-path acceptance strategy with a session-decoupled ultimatum phase, and a predictive wall-clock deadline guard to mitigate real-time computation forfeits under the 3-minute session cap.

1. Introduction

Autonomous negotiation agents in open environments face concurrent challenges: optimizing their own utility, predicting opponent behavior, and defending against strategic exploitation—such as tracking and reverse-engineering preference profiles. In ANL 2026, agents are evaluated based on a scoring function balancing self-advantage and trackability:

$$\text{score}(\mathbf{A}) = \text{adv}(\mathbf{A}) + \tau(\mathbf{A}) / (\tau(\mathbf{A}) + \tau(\mathbf{B}))$$

Where $\text{adv}(\mathbf{A})$ represents the advantage of agent $\mathbf{A}$, $\tau(\mathbf{A})$ is the trackability of agent $\mathbf{A}$, and $\tau(\mathbf{B})$ is the trackability of the opponent.

The file **ianos.py** optimizes this objective function through a dual-intent design: it aggressively suppresses its own trackability ($\tau(\mathbf{A}) \rightarrow 0$) via systematic rank inversion, while utilizing a highly resilient portfolio of modeling techniques to track the opponent ($\tau(\mathbf{B}) \uparrow$). Furthermore, it accounts for real-world execution constraints by monitoring per-round wall-clock compute overhead to avoid self-inflicted forfeits.

2. Concealing Bidding Strategy

The bidding subsystem of **ianos** aims to find agreements near our target utility space while completely obfuscating our underlying utility function from the opponent's learning algorithms. This is achieved through a four-tier concealment pipeline executed during the bidding phase.

2.1 Tier 1: Decoy Bids

- In the early game where the relative time $t < 0.55$ (DECOY_MAX_TIME), the agent stochastically emits decoy bids.
- These decoys are triggered with a probability of **12%** (DECOY_PROB).
- When a decoy is triggered, the targeted utility is artificially dropped by a fixed margin of **0.06** (DECOY_DROP).
- By injecting structured variance early on, Ianos disrupts the opponent's early-stage linear or concession-based modeling.

2.2 Tier 2: Target Jittering

- For standard bids, the true target utility calculated by the concession curve is perturbed using zero-mean Gaussian noise.
- The noise is parameterized with a standard deviation $\sigma = 0.018$ (JITTER_SIGMA).
- The perturbed target utility is tightly clipped at a maximum deviation of ± 0.035 (JITTER_CLIP).
- This ensures the long-run average target utility remains strategically sound while preventing the opponent from back-calculating our exact concession steps.

2.3 Tier 3: Poison Pill Rotation

- To pollute the opponent's frequency-based issue estimators, Ianos implements a deterministic, phase-shifted poison rotation mechanism.
- The agent targets the most vital issue spaces and forces a specific issue-value pair into the viable bidding pool.
- The issue slot rotates every **15 turns** (POISON_ROTATION_INTERVAL).

- The value slot rotates every **8 turns** (`POISON_VALUE_ROTATION_INT`).
- These frequencies are shifted using a session-unique random salt (`_poison_salt`) to ensure they remain unpredictable across mirror matches.

2.4 Tier 4: Enhanced Rank Inversion (CONCEAL 3)

Among the viable outcomes matching the jittered target utility within a tolerance window of ± 0.05 , **ianos** applies an obfuscated selection bias instead of uniform sampling. The agent ranks its true issue importance based on the marginal standard deviation of utilities across top outcomes:

- **Freeze Top Issues:** For the top N issues (`INVERSION_DEPTH = 4`), the agent biases selection *away* from its true modal value, making highly rigid issues appear flexible to an observer.
- **Vary Bottom Issues:** For low-value issues, it biases selection *toward* its true modal value, mimicking artificial rigidity on unimportant issues.
- **Stochastic Weighting:** Outcomes that successfully invert these true issue rankings are given a probability multiplier of **4** (`INVERSION_WEIGHT`), driving $\tau(A) \rightarrow 0$.

3. Robust Opponent Modeling

To accurately estimate the opponent's utility function, **ianos.py** uses a multi-model portfolio designed to filter out opponent decoys and generalize well across diverse domain types.

3.1 Recency-Weighted Frequency Estimator

- Instead of relying on standard cumulative frequency counts, Ianos tracks issue-value counts dynamically.
- It applies a recency multiplier of **3.0** (RECENCY_FACTOR) for bids observed late in the game when $t \geq 0.75$ (RECENCY_T).
- Issue weights are dynamically derived from tracking issue movement, where highly mobile issues are penalized and frozen issues are assigned higher importance:

$$w_i = 1 / (1 + m_i / m_{\max})$$

Where m_i is the weighted move count of issue i , and $m_{\max}$ is the maximum movement recorded across any issue space.

3.2 Robust MAD Outlier Filtering

- Estimated opponent utilities are passed through a rolling Median Absolute Deviation (MAD) filter window of size **10** (ROBUST_WINDOW) once a minimum of **5 bids** (ROBUST_MIN_WINDOW) have been observed.
- The MAD is computed as: $MAD = \text{median}(|x_i - \text{median}(x)|)$.
- If an incoming bid deviates by more than $3.0 \times MAD$ (ROBUST_MAD_THRESH) from the window median, it is flagged as an outlier or decoy attempt.
- Instead of omitting it completely, its weight in the frequency updating process is down-weighted by a factor of **0.15** (ROBUST_DOWNWEIGHT) to prevent tracking pollution while adapting to legitimate strategy shifts.

3.3 Lightweight Gaussian Process Concession Forecasting

To forecast the opponent's concession path, Ianos maintains a historical trajectory of time-utility pairs and fits a lightweight Gaussian Process regression model using a Squared-Exponential (RBF) kernel:

$$k(t1, t2) = \sigma_f^2 \times \exp(-(t1 - t2)^2 / (2 \times \ell^2)) + \sigma_n^2$$

Where $\ell = 0.15$ (GP_LENGTHSCALE) regulates smoothness, $\sigma_f^2 = 0.05$ (GP_SIGNAL_VARIANCE) governs output variance, and $\sigma_n^2 = 0.01$ (GP_NOISE_VARIANCE) provides observation noise tolerance. The system solves the linear system via a pure Python Gauss-Jordan implementation restricted to a maximum window of **25 observations** (GP_MAX_OBSERVATIONS) to control computational complexity.

3.4 Portfolio Selector and Archetype Classifier

At initialization, the agent evaluates the shape of the outcome space to assign its portfolio family:

- **SPARSE_DOMAIN:** Selected if outcomes ≤ 500 and issues ≤ 4 . The GP forecast blend weight (**PORTFOLIO_GP_BLEND_SPARSE**) is kept low at **10%** since raw frequency counts update quickly.
- **DENSE_DOMAIN:** Selected for larger spaces where value observations are sparse. The agent relies heavily on the GP-smoothed consensus score, blending it into the opponent utility estimation at **40%** (**PORTFOLIO_GP_BLEND_DENSE**).

Additionally, during the first **8 turns** (ARCHETYPE_WINDOW), an archetype classifier measures the average per-step concession velocity. Opponents are tagged as **FAST** (concession drop ≥ 0.020), **SLOW** (concession drop ≤ 0.004), or **MEDIUM**.

4. Acceptance Strategy and Ultimatum Phase

The acceptance strategy evaluates incoming offers through five distinct decision paths, prioritizing computational efficiency and strategic concession.

Summary of Decision Paths

- **Path 0 (Wall-Clock Danger Mode):** Triggered by the deadline guard. It bypasses all strategy logic to immediately accept any offer exceeding the reservation value ($u > r$).
- **Path A (Standard Target Compliance):** Accepts if the offer utility meets the current target modified by dither: $u_offer \geq u_target + dither$, where **dither** is sampled uniformly from ± 0.022 (ACCEPT_DITHER).
- **Path A* (Generous Offer Bypass):** Accepts if negotiation time $t \geq 0.50$ (SIMPLE_THRESHOLD_MIN_TIME) and $u_offer \geq 0.95 \times \max_u$ (SIMPLE_THRESHOLD).
- **Path B (Opponent Floor Extraction):** Accepts if $t \geq 0.72$ (FLOOR_ACCEPT_TIME), a stationary opponent floor is detected, and $u_offer \geq 0.78 \times \max_u$ (EXTRACTION_FLOOR).
- **Path C (Last-Resort Deadline):** Instigated if $t > 0.99$ and $u_offer \geq 0.60 \times \max_u$ (DEADLINE_FLOOR), provided the opponent is not flagged as stubborn.

4.1 Target Utility and Adaptive Panic Curve

The target utility curve remains highly conservative during the early negotiation phase, keeping demands near maximum utility. Once relative time passes the panic start threshold, the target utility begins tracking a non-linear concession path toward the extraction floor:

$$u_target = u_max - (u_max - u_floor) \times ((t - \text{panic_start}) / (1 - \text{panic_start}))^9$$

The **panic_start** variable is dynamically tuned based on the classified opponent archetype to maximize utility. For **FAST** opponents, it shifts to **0.96** (PANIC_START_FAST) to exploit their rapid concessions. For **SLOW** opponents, it drops to **0.85** (PANIC_START_SLOW) to initiate earlier concessions and avoid deadlocks.

4.2 Dynamic Ultimatum Phase

When negotiations reach the ultimatum trigger time, the agent freezes its concession path to maintain a strict bottom-line demand. To prevent concurrent freezes and perpetual deadlocks during mirror matches, the trigger time is randomized per session using a unique salt:

$$t_{\text{ultimatum}} = 0.95 + \epsilon_{\text{ult}}$$

Where ϵ_{ult} is sampled from ± 0.02 (ULTIMATUM_JITTER). The final ultimatum demand leaves a small, randomized incentive margin of $2\% \pm 1\%$ (EPSILON_INCENTIVE_BASE $\pm$ EPSILON_JITTER) above the extraction floor to encourage a last-second opponent acceptance.

Crucially, the ultimatum phase can be triggered **ahead of schedule** if the GP concession forecaster detects that the opponent's utility curve has flattened below a delta of **0.01** (GP_ULTIMATUM_PLATEAU_DELTA) across the forecast horizon. If a plateau is detected, the agent latches into the ultimatum phase immediately to pressure the stationary opponent rather than wasting rounds.

5. Wall-Clock Deadline Guard

A core innovation of **Ianos.py** is its defense against execution-time forfeits. While NegMAS's built-in relative time effectively balances standardized step counts, it cannot account for internal computation spikes caused by large-domain outcome scans, matrix system updates, or outlier filtering.

- **Measurement:** Ianos measures the real-world execution time between rounds at the very start of its execution block, capturing both the opponent's processing delay and its own algorithm overhead.
- **Smoothing:** It maintains a rolling window of **8 rounds** (WALLCLOCK_WINDOW) and builds an Exponentially Weighted Moving Average (EWMA) of the round cost:

$$C_{\text{avg}} = 0.7 \times C_{\text{avg}} + 0.3 \times \Delta t$$

- **Guard Trigger:** The deadline guard evaluates the system's remaining time against a conservative buffer:

$$\text{Safety Threshold} = 2.5 \times C_{\text{avg}} (\text{WALLCLOCK_SAFETY_MARGIN})$$

If the remaining session time drops below this threshold, the agent permanently locks into **Forced Acceptance Mode (Path 0)**. This terminates the expensive computation loops required for bidding and modeling, immediately accepting any incoming rational offer ($u_{\text{offer}} > \text{reserved_value}$). This protects the agent from self-inflicted execution forfeits, which score zero advantage points for both parties.

6. Conclusion

Ianos presents a balanced approach to automated negotiation under complex structural constraints. By pairing aggressive preference concealment (rank inversion and decoy scheduling) with robust modeling tools (MAD filtering, portfolio adjustments, and GP forecasting), the agent maintains a strong advantage while minimizing its trackability footprint. Guarded by an adaptive acceptance state machine and an active wall-clock deadline monitor, Ianos provides a highly resilient strategy for the competitive ANL 2026 circuit.

7. References

1. Hindriks, K. and Tykhonov, D. (2008). Opponent modelling in automated multi-issue negotiation using Bayesian learning. In Proceedings of AAMAS 2008.
2. Renting, B. M., Baarslag, T., and Jonker, C. M. (2020). Combining manual and automated strategies in bilateral negotiations. In Proceedings of AAMAS 2020 Workshop on Agent-Mediated Electronic Commerce.
3. Renting, B. M., Baarslag, T., and Jonker, C. M. (2022). A portfolio meta-learning approach for automated negotiation. In Proceedings of IJCAI 2022.

4. Tunali, O., Aydoğan, R., and Sánchez-Anguix, V. (2017). Rethinking frequency opponent modeling in automated negotiation. In Proceedings of PRIMA 2017.

5. Williams, C. R., Robu, V., Gerding, E. H., and Jennings, N. R. (2011). Iamhaggler2011: A Gaussian process regression based negotiation agent. In Proceedings of the IJCAI 2011 Workshop on Agent-Based Complex Automated Negotiations.