

EmSel: A Robust Time-Dependent Agent for SCML Standard League

Selen Recepoglu, S018282, selen.recepoglu@ozu.edu.tr
Emre Karaarslan, S015174, emre.karaarslan@ozu.edu.tr

Abstract

This report describes EmSel, our negotiation agent built for the Standard track of the ANAC 2026 Supply Chain Management League (SCML). Built on the `StdSyncAgent` interface, it uses the classic Bidding–Opponent Modelling–Acceptance architecture. Crucially, every offer is sized to the factory’s actual needs, allowing aggressive pricing without risking unfulfillable contracts. We integrated a light opponent model (exponential moving average plus a rough persona) to tune concession rates, and a signing rule that commits only to what the factory can absorb. Heavy emphasis was placed on robustness; after fixing NegMAS/SCML API integration issues, EmSel runs flawlessly without crashing. Over five local tournaments against built-in baselines, it averaged 0.738, won two runs outright, placed a close second in the others, and never finished with a negative score.

1 Introduction

Automated negotiation is central to multi-agent systems, where independent agents must reach deals without centralized coordination [1]. SCML places this problem inside a simulated economy where agents buy inputs, produce goods, and sell outputs [2]. The Standard track adds complexity via inventory carry-over and negotiated delivery dates. Furthermore, agreements are only binding once both sides sign them, turning contract signing into a critical strategic decision.

Our aim with EmSel was to build a highly robust agent based on rational decision-making under time pressure [3]. Our guiding principle was survival: an agent must avoid bankruptcy and, just as importantly, runtime crashes. This report details our negotiation strategy, the engineering required for API stability, and performance results against league reference agents.

2 Background and Related Work

Time-dependent negotiation behavior often traces back to Faratin et al., treating offers as a concession function over time [5]. An agent can anchor

high and concede slowly, accelerating only near the deadline (a Boulware strategy) [6]. EmSel adopts this approach for pricing.

However, the Standard game rewards smart trading decisions alongside haggling. Since over-committed contracts incur shortfall penalties [2], simply maximizing volume leads to financial loss, especially against unpredictable opponents [4]. Therefore, we tied negotiation targets directly to inventory needs. We maintained the standard Bidding–Opponent Modelling–Acceptance structure [6] but added a strict contract-signing step tailored for the Standard track.

3 Methodology

EmSel subclasses `StdSyncAgent`, answering all daily offers simultaneously in `counter_all` and setting openings in `first_proposals`. We prioritized a simple, predictable decision rule to ensure the agent stays within its per-step time budget and avoids environment-induced crashes.

3.1 Bidding: Needs-Aware Aspiration

For pricing, EmSel opens at the most favorable legal limit (highest when selling, lowest when buying), anchoring the negotiation. We determine our role dynamically from the negotiation annotation rather than static attributes, as roles can flip in deep supply chains.

Unlike standard aspiration agents that demand maximum quantities, EmSel strictly requests only its unmet need:

$$q^* = \text{clip}(\text{need}, q_{\min}, q_{\max}) \quad (1)$$

It reads this from `awi.needed_supplies` or `awi.needed_sales`. This identical cap applies to counters, ensuring we never request unusable units. This single modification drastically reduced losses from bad trades (buying unneeded inputs or over-promising outputs).

3.2 Opponent Modelling: EMA Trend and Personas

To predict price directions efficiently, EmSel tracks the offer history from each partner, $H = \{p_1, \dots, p_k\}$, smoothing it with an exponential moving average ($\alpha = 0.6$):

$$\text{EMA}_k = \alpha p_k + (1 - \alpha) \text{EMA}_{k-1} \quad (2)$$

This is mixed with a short linear trend to mitigate the impact of outliers:

$$P_{\text{pred}} = \frac{[0.7p_k + 0.3(p_k - p_{k-1})] + \text{EMA}_k}{2} \quad (3)$$

EmSel also assigns a rough persona based on average concession step sizes. Partners conceding under 0.2 are **STUBBORN** (we hold firm early, easing near the deadline). Those conceding over 1.2 are **CONCEDERS** (we raise our bar to extract margin). Others default to **BALANCED**.

The counter-offer is a gentle adjustment from the prediction, capped within legal bounds:

$$p_{\text{counter}} = \begin{cases} \max(p_{\text{recv}}, 1.02P_{\text{pred}}) & \text{as seller,} \\ \min(p_{\text{recv}}, 0.98P_{\text{pred}}) & \text{as buyer.} \end{cases} \quad (4)$$

We limited this push to $\pm 2\%$ to prevent stalling negotiations.

3.3 Acceptance: Time, Inventory, and Sniping Guard

EmSel evaluates offers against a dynamic utility threshold that starts at 0.80 and drops linearly with normalized time $t \in [0, 1]$, adjusted by persona, quantity, and inventory terms:

$$U_{\text{th}} = \max\left(0.50, 0.80 - 0.22t + \Delta_{\text{adaptive}} + \Delta_{\text{quantity}} + \Delta_{\text{inventory}}\right) \quad (5)$$

The inventory term allows the factory’s physical state to influence negotiations. When buying with critically low input stock (≤ 2), we lower the bar by up to 0.16 to prevent line stalls. When selling with excess output (≥ 10) near the deadline, we lower it by up to 0.14 to free up storage. Finally, a sniping guard activates when $t > 0.95$, accepting anything within 0.04 of the threshold to secure last-minute viable deals.

3.4 Contract Signing

Because Standard track contracts only bind when signed, we override `sign_all_contracts`. EmSel sorts pending buys (cheapest-first) and sells (dearest-first), signing down each list until factory needs are met. Excess contracts remain unsigned. As a fail-safe, if needs read as zero or cannot be determined, we fall back to the framework’s default behavior. This prevents accidental starvation while trimming harmful over-commitments.

4 Implementation and Robustness

A significant portion of development involved correctly interfacing with the NegMAS/SCML API. Several critical fixes ensured perfect stability:

- **Response Construction:** Replaced incorrect `SAOResponse` attribute calls with `ResponseType` enumeration constants (e.g., `ResponseType.REJECT_OFFER`).

- **Offer Representation:** Handled SCML offers as positional tuples. Accessing prices via index rather than string names was necessary to successfully activate our EMA logic.
- **Mechanism Safety:** Wrapped `get_nmi(partner_id)` in `try/except` blocks and checked for `None` before accessing `.issues`, preventing dead negotiations from crashing the system.

With these guards, EmSel completed every world tested. (Note: Baseline agents like `SyncRandomStdAgent` occasionally threw internal `NoneType` errors during our tests, but EmSel’s safety checks allowed the tournament to proceed and score normally).

5 Results

We tested EmSel against the Standard baseline (`SyncRandomStdAgent`) and `RandomOneShotAgent` across five diverse local tournaments (see Table 1).

Table 1: Tournament scores. EmSel won two runs, placed a close second in the rest, and remained the only agent with strictly positive scores.

Run	EmSel	SyncRandomStd	RandomOneShot
1	0.727	0.579	0.206
2	0.954	1.064	0.733
3	0.503	0.603	−0.063
4	0.958	0.988	0.002
5	0.548	0.513	0.263
Mean	0.738	0.749	0.228
Min	0.503	0.513	−0.063

EmSel’s average (0.738) is highly competitive with the baseline (0.749). More importantly, it demonstrated exceptional consistency: its minimum score was 0.503, while the one-shot agent dipped into negatives. EmSel secured two outright victories and consistently avoided bankruptcy.

6 Discussion and Conclusions

Our results confirm that in the SCML Standard game, strict inventory alignment out-performs sheer volume or maximum price squeezing. By capping requests to actual `needed_supplies` and `needed_sales`, EmSel drastically improved its viability. Furthermore, its cautious acceptance threshold and targeted signing rule ensured it almost never locked into losing contracts, proving that survival and trading discipline are paramount.

EmSel successfully combines aggressive pricing, needs-aware sizing, lightweight opponent modeling, and a hardened API implementation. Its main weakness is variance, caused by a rigid concession schedule (evident in the gap between Run 3 and Run 4). For future work, we plan to implement a highly adaptive, learned concession schedule and expand our needs-aware logic to coordinate multiple input/output streams simultaneously, addressing the deeper complexities of multi-tier supply chains.

References

- [1] Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons.
- [2] Nishino, N., et al. (2020). “Supply Chain Management League: A New Challenge for ANAC.” *Proceedings of the 13th International Workshop on Agent-based Complex Automated Negotiations*.
- [3] Jennings, N. R., et al. (2001). “Automated negotiation: prospects, methods and challenges.” *International Journal of Collective Intelligence*.
- [4] Kraus, S. (2001). *Strategic Negotiation in Multiagent Environments*. MIT Press.
- [5] Faratin, P., et al. (1998). “Negotiation decision functions for autonomous agents.” *Robotics and Autonomous Systems*, 24(3-4), 159-182.
- [6] Baarslag, T., et al. (2014). “Decoupling Negotiating Agents to Explore the Space of Negotiation Strategies.” *Studies in Computational Intelligence*, vol 535, 61-83. Springer.