

StdOmegaNegotiator: Quantity-First Bounded Optimization for SCML 2026 Standard Track

Genchan Omega
SCML 2026 Standard Track Submission

Abstract

We present **StdOmegaNegotiator**, an autonomous factory manager for the SCML 2026 Standard Track. The agent coordinates concurrent supply and sales negotiations using an explicit daily production target, inventory-aware quantity planning, bounded dynamic programming for same-day offer selection, near-future contracting, time-based price concession, and lightweight partner reliability estimates. Its main design objective is to maintain production flow without the exponential powerset search used by an earlier prototype. Across development and the final deadline audit, we evaluated more than seventy strategy configurations using common random numbers and same-tournament comparisons against the top three Standard agents from SCML 2025. The adopted architecture improved mean normalized score from 0.9625 to 1.1221 over its predecessor in the primary direct comparison.

1. Introduction

The SCML Standard Track couples bilateral negotiation with multi-day production and inventory management. A contract can have an attractive unit price but still reduce final profit if its quantity cannot be processed, if matching input or output is unavailable, or if it creates excessive inventory. In addition, a factory negotiates concurrently with several partners, so accepting offers independently can oversupply one side of the production process.

Our first agent addressed this coupling with utility-based bundle search built on a synchronized random baseline. In public Standard tournament #19178, started on June 20, 2026, the downloaded legacy submission ranked 19th with score 6,275. The agents had been downloaded on June 19 at 20:20 UTC, before the quantity-first implementation documented here was finalized. Local validation also showed that merely updating the old architecture did not consistently improve it. We therefore replaced the negotiation core rather than continuing local parameter tuning.

The final design is inspired by the quantity-first control used by successful prior Standard agents, but it is implemented independently with bounded algorithms and no dependency on **scml-agents**. The submitted class is **StdOmegaNegotiator** in module `submission.std_agent.entry`.

2. Production and Quantity Planning

Let L be the number of production lines, I the current input inventory, and $p = 0.75$ the tuned production target. For day t , the target production quantity is $P = \text{round}(pL)$. The remaining supply need is

$$N_{\text{buy}}(t) = \max\{0, P - I - S(t)\},$$

where $S(t)$ is already contracted input for day t . The remaining sales need is

$$N_{\text{sell}}(t) = \max\{0, \min(L, P + I) - D(t)\},$$

where $D(t)$ is already contracted output. These equations are recalculated before every synchronous response, including quantities accepted in the current response.

For initial and counter proposals, the agent normally activates the best-ranked 75% of available partners, while ensuring that their combined quantity capacity can cover the need. Late in the simulation this fraction increases by 15 percentage points to improve completion. If the recent balance trend is falling, it decreases by 10 points to limit risky commitments. Quantities are distributed evenly subject to each negotiation's maximum quantity.

3. Negotiation Strategy

3.1 Same-day offer selection

Current-day buying and selling offers are processed separately because each side has its own required quantity. For one side, the agent builds a dynamic-programming table indexed by total quantity. Each state stores the highest-quality subset found for that quantity, where quality combines total price and a small partner-reliability term. The table is bounded at

$$N + \max\{\tau, q_{\max}\},$$

where N is the current need, τ is a small adaptive overfill threshold, and $q_{\max}$ is the largest candidate offer. The selected state is the smallest permitted overshoot; if none exists, it is the closest positive underfill. Price quality breaks quantity ties.

This method captures useful combinations without enumerating 2^n subsets. Its state space is bounded by feasible daily quantity rather than the number of partners, which keeps response time stable in large negotiations. The implementation does not call the utility function once per subset and contains no print statements.

3.2 Future contracts

Unused partners receive proposals for the next three days. Ranked partners are split approximately 50%, 30%, and 20% across offsets one, two, and three. Each future day’s proposal quantity is one third of its estimated need. Incoming future offers are considered in chronological order, with favorable prices first, and accepted only while cumulative quantity remains below 95% of estimated need. This slight reserve reduces overcommitment while keeping the production pipeline active.

3.3 Price concession

First proposals use the most favorable issue-bound price: the minimum when buying and the maximum when selling. Counteroffers concede smoothly with negotiation-relative time r :

$$c(r) = 0.25 + 0.75r^{1.7}.$$

The buying price moves from the minimum toward the maximum by fraction $c(r)$; the selling price moves in the opposite direction. An additional late-stage concession is applied when unmet needs are urgent. Acceptance uses a configurable concession bound; the selected policy prioritizes quantity feasibility and accepts the full valid issue-price range, with price used to choose among equally fitting bundles.

3.4 Partner reliability

The opponent model records actual negotiation successes and failures through SCML callbacks. A partner’s reliability is its empirical success rate, initialized neutrally at 0.5. Reliability orders proposal recipients and provides a bounded tie-break in subset selection. One exploration partner may be sampled outside the preferred group so that early failures do not permanently exclude a potentially useful counterparty.

4. Optimization Procedure

We first tested 17 modifications to the previous architecture, including partner caps, future horizons, price memory, phase-dependent acceptance, and bounded cross-side seeds. A promising cross-side search result failed a same-tournament direct test, winning only one of three seeds, so it was rejected.

We then implemented the new quantity-first architecture and screened 13 configurations. To control world-generation noise, every configuration used the same candidate class name, the same two seeds, and the same opponents: `AS0`, `XenoSotaAgent`, and `SyncRandomStdAgent`. The selected configuration used productivity 0.75, partner fraction 0.75, and future acceptance ratio 0.95. Its screening mean was 1.1140, with a positive mean advantage of 0.1202 over the fixed opponent pool.

For final validation, the selected agent and previous agent participated in the same tournament worlds against `AS0`, `XenoSota`, `Ultra`, and `SyncRandom`. Table 1 reports normalized scores.

The selected agent won all three paired comparisons. A final archive-level smoke tournament using pinned SCML 0.8.2 and NegMAS 0.15.5 also completed successfully. `StdOmegaNegotiator` scored 0.9375 versus 0.8311 for `GreedyStdAgent` and 0.8070 for `SyncRandomStdAgent`.

Table 1: Same-tournament direct validation.

Seed	Previous agent	StdOmegaNegotiator	Difference
260620	1.0387	1.1590	+0.1203
260621	0.9656	1.0399	+0.0743
260622	0.8833	1.1674	+0.2841
Mean	0.9625	1.1221	+0.1596

4.1 Final deadline audit

Immediately before final submission, we tested 45 additional configurations in three stages. The first stage varied output-inventory handling, official need estimates, partner learning, future quantities, future price filters, level-specific productivity, and margin guards. The second stage evaluated 16 opening, counteroffer, and acceptance-price policies. The third stage tested 10 combinations near the apparent price-policy winners. Screening used stable class names and common random numbers; finalists were then placed in the same tournament as the current agent and the 2025 top three.

Table 2 summarizes the decisive direct tests. Positive values favor the candidate. A no-learning policy and combined price policy failed clearly. Counter concession 0.20 and first-offer concession 0.10 showed small gains in one context, but both had losing seeds and failed to reproduce in the final five-seed tournament. We therefore retained the pre-audit policy rather than selecting noise.

Table 2: Final held-out direct validation of proposed changes.

Candidate	Mean difference	Wins	Decision
Disable partner learning	-0.0069	0/3	Reject
Counter concession 0.20	+0.0076	5/8	Reject: unstable
First-offer concession 0.10	+0.0034	6/8	Reject: unstable
Combined price policy	-0.0174	0/5	Reject

5. Discussion

The experiments produced three practical conclusions. First, architecture choice mattered more than further tuning of the previous hybrid. Second, common random numbers and same-tournament direct comparisons were necessary because independently generated tournament assignments produced misleading apparent gains. Third, quantity-oriented control generalized better than expensive utility enumeration in our tests. The final audit also confirmed that a small screening advantage is not enough evidence for a deadline submission: every apparent last-minute improvement failed at least one stronger direct test.

The current opponent model is intentionally small. Future work could estimate partner-specific agreement times and price distributions, or use different production targets for first, middle, and last production levels. Such extensions should be validated with paired worlds to avoid selecting configuration noise.

Conclusion

`StdOmegaNegotiator` combines inventory-aware daily planning, bounded quantity-DP acceptance, near-future contracting, time-based concession, and partner reliability. It replaces an unstable and computationally heavier predecessor and remains compatible with the pinned SCML submission environment. The final policy is the most extensively validated version, not the highest single-run configuration.