

OneshotOmegaNegotiator: Scalable Quantity–Price Portfolio Negotiation for SCML 2026 OneShot

Genchan Omega
SCML 2026 OneShot Submission

Abstract

We present `OneshotOmegaNegotiator`, our submission to the SCML 2026 OneShot track. The agent treats concurrent offers as a contract portfolio rather than independent bilateral decisions. Candidate bundles are ranked by a quantity–price score and the official SCML utility function. Exact subset search is used for at most ten partners, while a bounded beam search prevents exponential growth in larger markets. The policy combines a time-dependent acceptance threshold, equal quantity distribution, a small buyer-side fulfillment buffer, an adaptive per-partner capacity cap, market-aware deterministic prices, and Bayesian partner reliability estimates. Across the final 48 unseen mixed and strong test worlds, the final strategy preserved mixed-environment performance and improved the strong-opponent mean without changing seller performance.

1. Introduction

The SCML OneShot track differs from classical bilateral negotiation settings in one essential respect: the utility of any contract depends strongly on the other contracts concluded on the same day. A cheap supply contract is not beneficial if the resulting inventory cannot be converted into profitable sales. Likewise, an apparently attractive sell contract can become costly if the corresponding inputs cannot be secured without triggering shortfall penalties. Because of this coupling, local offer-by-offer reasoning is often insufficient.

Our design objective was therefore to build an agent that reasons globally across simultaneous negotiations while still responding within the strict timing regime of the SCML simulator. We focused on three concrete goals:

1. maximize daily utility under the official profit function;
2. reduce losses from over-commitment, shortfall, and low-value late agreements;
3. remain robust against a diverse set of unknown opponents without requiring heavy offline learning.

2. Overall Design

The final submission is based on a synchronous controller implemented on top of `OneShotSyncAgent`. At each decision point, the agent receives the current offers from all running negotiations and responds jointly. This makes it possible to coordinate acceptance and counteroffers across partners instead of treating each bilateral interaction independently.

The submitted module is `submission.oneshot_agent.entry`. Its public class `OneshotOmegaNegotiator` wraps the final `FinalAdaptiveCapAgent` strategy described below.

The architecture has five tightly coupled components:

1. **bundle evaluation:** candidate subsets of simultaneous offers are scored using the SCML utility function;
2. **quantity–price acceptance:** a time-sensitive score threshold determines whether the best available bundle is good enough to accept;
3. **price targeting:** counteroffers are generated from utility-based reservation prices, public market signals, and opponent-specific anchors;
4. **quantity allocation:** required quantity is distributed across open negotiations in a conservative way to reduce simultaneous over-acceptance risk;
5. **bounded search:** large concurrent offer sets use beam search rather than an uncapped powerset.

This architecture is deliberately rule-based. Our experiments showed that in the OneShot setting, careful coordination of utility, quantity, and timing provides more reliable gains than introducing heavier learning components without enough per-negotiation data.

3. Bundle-Based Acceptance Strategy

The core of the agent is a bundle-based acceptance strategy. For a candidate bundle B , let $q(B)$ be its total quantity and n the remaining operational need. The quantity component is $Q(B) = 1 - |q(B) - n| / \max(1, n)$. A normalized price component includes total contract value and role-specific shortfall or disposal penalties. The primary score is $0.85Q(B) + 0.15P(B)$; official utility, remaining shortfall, and bundle size are used as tie-breaks.

The acceptance threshold starts at 0.60 and decreases linearly to 0.30 over negotiation time. This favors near-complete portfolios early while allowing partial but useful agreements near the deadline. For ten or fewer current partners all subsets are evaluated. Above ten, offers are ordered by their individual utility and reliability and explored with a bounded beam, avoiding the uncapped 2^n behavior that can stall large OneShot worlds.

4. Price Generation

Counteroffer prices are determined from three information sources.

Utility-based reservation price. At the start of each day, the agent probes the utility landscape to estimate a reservation-like price level for one-unit agreements. This avoids relying only on the raw issue bounds.

Public market information. The agent reads the current trading price and exogenous contract summary and derives a simple market-pressure signal from them. This allows the pricing policy to react differently in relatively supply-rich and demand-rich situations.

Partner-specific anchors. For each counterpart, the agent stores simple statistics such as first observed price, best observed price, most recent price, average successful price, and a concession trend score. These values influence both the price target and the quantity priority given to that partner.

The resulting policy does not make offers at a fixed concession schedule only. It instead moves over time between an aggressive price and a utility-grounded target while remaining influenced by both market information and observed opponent behavior.

5. Quantity Allocation and Risk Control

In OneShot, quantity is at least as important as price. A strategy that ignores quantity coordination can accept a seemingly attractive set of deals and still perform poorly because of production constraints or poor downstream matching.

Our quantity allocator computes the remaining need from the current world state and distributes it nearly equally across active negotiations. Buyers request 10% more than the current need before integer rounding because uncovered input tends to incur a stronger shortfall cost; sellers use the exact remaining need. The default per-partner cap is three units. If active partners cannot collectively cover the target, the cap is raised to the ceiling of target quantity divided by active partner count. Several safeguards are built into the policy:

- issue minimum and maximum quantities are respected explicitly;
- the three-unit cap is raised only when partner scarcity makes fulfillment otherwise impossible;
- simultaneous acceptance is selected as one portfolio, preventing independent over-acceptance;
- Bayesian success/failure estimates break ties between otherwise similar partners.

These mechanisms significantly reduced two common failure modes observed in early versions of the agent: contracting too much volume too early, and continuing unproductive negotiations too long near the deadline.

6. Opponent Model

The opponent model is intentionally lightweight. We avoid heavy inference methods because each negotiation is short and per-opponent data is limited. Instead, for each partner we maintain:

- the first, best, and latest observed prices;
- the average price of successful agreements;
- counts of successes and failures;
- an exponentially smoothed concession score.

This information is sufficient to distinguish cooperative partners from stubborn ones and to bias quantity and pricing decisions accordingly. The concession score is particularly useful because it captures directional movement rather than only absolute price level.

7. Development and Evaluation

The final version was obtained through more than twenty diagnose–implement–evaluate cycles. The most important improvements were:

1. correcting the synchronous response flow so that all rejected bundles still receive valid counters;
2. enforcing quantity issue bounds more carefully;
3. fixing evaluation with per-world common random numbers, equal buyer/seller positions, and a common evaluation class name;
4. integrating quantity and price into one portfolio score rather than using utility aspiration alone;
5. replacing random counter prices with deterministic market and partner targets;
6. tuning buyer over-ordering to 10% and capping exhaustive search at ten partners;
7. recovering proposal capacity dynamically when too few partners remain.

We evaluated the old and final agents on identical generated worlds. The mixed pool contained Greedy, EqualDist, RandDist, and Random agents. The strong pool also contained the public 2024/2025 Cautious, AlmostEqual, Rchan, and CostAverse agents. Final unseen tests used six new seeds, four worlds per seed, 20 simulation steps, and balanced buyer/seller placement. Mean scores were:

- mixed pool: final 1.115860, previous 1.115860, no regression;
- strong pool: final 1.073704, previous 1.071230, paired gain +0.002474.

In the strong test, buyer mean increased from 1.072389 to 1.077337 while seller mean remained exactly 1.070071. An additional scarce-partner test with four agents per process improved from 1.085749 to 1.086735.

8. Lessons and Future Work

Four lessons were especially important during development.

1. **Global coordination matters more than isolated bargaining heuristics.** The largest gains came from bundle-aware acceptance rather than from more complicated per-opponent models.
2. **Mechanisms must be evaluated jointly.** Over-ordering or quantity-first ranking alone did not help; the integrated quantity–price score with deterministic pricing did.
3. **Buyer and seller risk is asymmetric.** The buyer-side quantity buffer produced most of the validated gain, while larger 12–20% buffers were harmful.
4. **Capacity limits should depend on partner scarcity.** A fixed cap is safe in broad markets but prevents fulfillment when only a few partners remain.

Promising future work includes learned beam ordering, stronger within-simulation learning, and richer reliability models for price movement and timely agreement.

Conclusions

OneshotOmegaNegotiator combines bounded portfolio search, quantity–price acceptance, market-aware deterministic pricing, asymmetric quantity control, adaptive proposal capacity, and lightweight partner modeling. It avoids exponential search and improved strong-opponent performance in unseen paired tests over the version ranked eighth in the latest public mock tournament.