

LatticeOneShotAgent: ANAC 2026

Submitted Agent for the SCML OneShot Track

Yuzuru Kitamura
Tokyo University of Agriculture and Technology

June 22, 2026

1. Overview

The overall design of LatticeOneShotAgent is based on conservative quantity-risk control. The agent first computes a target quantity, then decides how much of that target to allocate to each partner and how to respond to incoming offers. The same basic principle is used on both the buying and selling sides. However, the effects of shortage and surplus differ depending on the market side, so the thresholds and the treatment of overordering are asymmetric.

The following sections describe the two main modules: first proposal and counter proposal.

2. First Proposal Strategy

2.1. Quantity target

At the beginning of each negotiation step, LatticeOneShotAgent computes the quantity it wants to secure on the current market side. On the selling side, this target is the current remaining sales need. On the buying side, the agent adds a modest overordering margin because shortage may be more costly than extra inventory.

The base coefficient of the overordering margin in the current buying-side first proposal is 0.25. Using this coefficient, the final overordering margin is defined as follows.

$$\text{overordering_margin} = \text{overordering_base} * \sqrt{\text{sd-ratio}}$$

Here, sd-ratio is an index used in CostAverseAgent, one of the previous winners, and represents the ratio between the magnitude of the shortfall penalty and the disposal cost. It is given by

$$\text{sd-ratio} = \frac{\text{shortfall_loss}}{\text{disposal_loss}}$$

where `shortfall_loss` and `disposal_loss` denote the magnitude of the effect of one unit of shortage and one unit of surplus, respectively, on the obtained utility.

Thus, the more costly shortage is estimated to be relative to disposal, the more aggressively the agent buys.

2.2. Partner filtering

The agent does not propose to bankrupt partners. In the later part of the world, it also avoids partners with no previous agreement history. The late phase begins after the earlier of half of the total number of world steps or step 60. This rule is intended to avoid wasting proposals on partners that have shown no evidence of successful trading.

This filtering is intentionally simple. Instead of precisely ranking all partners by long-term utility, the agent removes clearly weak proposal targets so that the allocation logic can focus on active partners with plausible trading opportunities.

2.3. Initial allocation before learning is used

For the first 25 steps, LatticeOneShotAgent uses legacy distribution logic rather than the learned first-proposal optimizer. This warmup period is mainly used to collect partner response data. If the learned model is used when there are almost no observations, the prior has an excessive influence, so the agent starts with a more stable balanced policy.

2.4. Learned first-proposal allocation

2.5. Three-state response learning in first proposals

LatticeOneShotAgent first determines the quantity that should be covered by first proposals on each negotiation day. Let P be the set of negotiable partners and N be the current required quantity. Here, N represents the quantity to be sold when the agent is a seller and the quantity to be procured when the agent is a buyer. When multiple partners are available, the agent proposes a total quantity that is slightly larger than the required amount. Let this total proposed quantity be

$$Q = \lfloor N(1 + \rho) \rfloor$$

where ρ is the over-ordering ratio, determined by the current setting. The first-proposal problem can therefore be formulated as an integer optimization problem: how to allocate the total proposed quantity Q among negotiation partners. If q_i denotes the proposed quantity to partner i , the problem is to select an allocation q satisfying

$$q = (q_i)_{i \in P}, \sum_{i \in P} q_i = Q, 0 \leq q_i \leq L$$

where L is the number of production lines and is used as an upper bound on the quantity proposed to a single partner. Rather than simply distributing the quantity evenly, LatticeOneShotAgent evaluates multiple candidate allocations and selects the one with the highest expected value according to a learned partner-response model.

For each partner i , negotiation side $s \in \{\text{buy}, \text{sell}\}$, and quantity bucket $b(q)$, the agent records past first-proposal outcomes as one of three states:

$$y_i \in \{A, C, R\}.$$

Here, A means that the offer was accepted as proposed and became a contract, C means that the offer was not accepted but the partner returned a counter offer, and R means that the negotiation ended. The quantity bucket $b(q)$ is a coarse discretization of the proposed quantity, for example intervals such as 1, 2, 3–4, 5–7, and 8 or more. This allows the agent to learn partner response tendencies over nearby quantity ranges rather than for each exact quantity, which makes decision-making more stable even with a small number of samples.

Let the numbers of past observations for partner i , side s , and quantity bucket b be $n_{i,s,b}^A$, $n_{i,s,b}^C$, and $n_{i,s,b}^R$. In the early part of a simulation, the number of observations is small, so directly using empirical frequencies can easily lead to overfitting. Therefore, the agent smooths the estimates using the prior distribution

$$\pi^A = \pi^C = \pi^R = \frac{1}{3}$$

and prior weight α . If the estimated probabilities of the three states are denoted by $p_{i,s,b}^A$, $p_{i,s,b}^C$, and $p_{i,s,b}^R$, they are defined as

$$p_{i,s,b}^A = \frac{n_{i,s,b}^A + \alpha\pi^A}{n_{i,s,b}^A + n_{i,s,b}^C + n_{i,s,b}^R + \alpha},$$

$$p_{i,s,b}^C = \frac{n_{i,s,b}^C + \alpha\pi^C}{n_{i,s,b}^A + n_{i,s,b}^C + n_{i,s,b}^R + \alpha},$$

$$p_{i,s,b}^R = \frac{n_{i,s,b}^R + \alpha\pi^R}{n_{i,s,b}^A + n_{i,s,b}^C + n_{i,s,b}^R + \alpha}.$$

A key feature of this model is that it does not merely estimate whether an offer will be accepted. Instead, it distinguishes acceptance, counter offers, and rejection. A counter offer is not an immediate contract, but it preserves the possibility of recovering the remaining shortage in later negotiation rounds. By contrast, rejection removes the recovery opportunity from that partner. Therefore, it is important not to treat counter offers and rejections as the same outcome.

Given a candidate allocation q , let $I(q) = \{i \in P : q_i > 0\}$ be the set of partners that receive a positive proposed quantity. For this allocation, partner responses are approximated as conditionally independent. Therefore, the probability that a response vector $y = (y_i)_{i \in I(q)}$ is realized is $P(y | q) = \prod_{i \in I(q)} p_{i,s,b(q_i)}^{y_i}$. Under response vector y , the quantity immediately secured is $A(y, q) = \sum_{i \in I(q): y_i = A} q_i$, and the total quantity remaining in continuing negotiations through counter offers is $C(y, q) = \sum_{i \in I(q): y_i = C} q_i$.

Let $U(x)$ be the SCML utility obtained when x units are immediately secured at the current first-proposal price. In evaluating a first proposal, however, the agent considers not only the utility of the immediately accepted quantity, but also the risk of excess contracts and the continuation value of counter offers. Let the difference from the required quantity be $\Delta(y, q) = A(y, q) - N$.

Let $\chi_+(z)$ be the indicator function that equals 1 when $z > 0$ and 0 otherwise, and let $\chi_-(z)$ be the indicator function that equals 1 when $z < 0$ and 0 otherwise. Then the score $S(y, q)$ for response y is represented as

$$S(y, q) = U(A(y, q)) - \lambda\kappa(0.5 + \Delta(y, q))\chi_+(\Delta(y, q)) + \eta\kappa \min(-\Delta(y, q), 0.5C(y, q))\chi_-(\Delta(y, q)).$$

The first term is the base utility obtained from the immediately accepted quantity. The second term is the penalty for excess contracts when the accepted quantity exceeds the required quantity. Since a first proposal becomes fixed as a contract once it is accepted, excessive acceptance may lead to disposal or unnecessary procurement. Therefore, a penalty is imposed when the excess $\Delta(y, q)$ is positive. The third term is the continuation value of having counter offers remain when the agent is still short. Even if the immediately accepted quantity is below the required amount, partners who returned counter offers may still lead to contracts in later rounds, so part of the shortage is evaluated as potentially recoverable.

Here, λ is the penalty coefficient for excess contracts, and η is the continuation-value coefficient for counter offers. In addition, κ is a unit coefficient used to adjust the utility scale. It is computed from local changes in utility around the current need as

$$\kappa = \max(|U(N) - U(N-1)|, |U(N) - U(N+1)|, p, \varepsilon)$$

so that even when the absolute scale of utility differs across worlds, the excess-contract penalty and the continuation value of counter offers are evaluated on a natural utility scale.

Thus, the expected score of a candidate allocation q is

$$V(q) = \sum_{y \in Y(q)} P(y | q) S(y, q),$$

where $Y(q)$ is the set of all three-state response vectors that can be produced by partners receiving positive quantities under allocation q . `LatticeOneShotAgent` selects the allocation that maximizes

this expected value. That is, if C_Q is the candidate allocation set, the selected first-proposal quantity allocation q^* satisfies

$$q^* \in C_Q, V(q^*) \geq V(q) \text{ for all } q \in C_Q.$$

In the implementation, this expected value can be computed either by explicitly enumerating all response combinations or by dynamic programming over aggregated states. If the number of partners receiving positive quantities is m , explicit enumeration evaluates up to 3^m combinations. By contrast, the dynamic-programming method keeps the joint distribution over the already accepted quantity a and the quantity remaining as counter offers c . The initial state is

$$D_0(0, 0) = 1$$

and partners receiving quantity q_i are processed one by one. The distribution is updated as follows:

$$D_{k(a+q_i, c)} \leftarrow D_{k(a+q_i, c)} + D_{k-1}(a, c)p_{i, s, b(q_i)}^A,$$

$$D_{k(a, c+q_i)} \leftarrow D_{k(a, c+q_i)} + D_{k-1}(a, c)p_{i, s, b(q_i)}^C,$$

$$D_{k(a, c)} \leftarrow D_{k(a, c)} + D_{k-1}(a, c)p_{i, s, b(q_i)}^R.$$

After all partners have been processed, applying the same score function S to each aggregated state (a, c) gives the expected value of allocation q . This method avoids explicitly enumerating every response combination and instead retains only the aggregated information needed for the decision: the accepted quantity and the quantity remaining in counter-offer continuation.

Finally, the selected allocation q^* is converted into first proposals

$$o_i = (q_i^*, t, p_i),$$

where t is the current negotiation day and p_i is the offered price. Under the current setting, the main adaptive component of first proposals is the quantity allocation. The three-state response model determines which partners should receive larger quantities and which partners should receive smaller or no proposals. With this design, LatticeOneShotAgent generates first proposals that simultaneously consider the probability of immediate contracts, the future recovery potential through counter offers, and the risk of excess contracts. The proposed price is uniformly set to the price most favorable to the agent.

3. Counter Proposal Strategy

3.1. Candidate Offer-Set Scan

In the counter-proposal stage, the agent first scans subsets of the incoming offers and searches for combinations that can be accepted as they are. Let P_c be the set of partners currently negotiating, x_i be the incoming quantity from partner i , and N be the current need. For an accepted subset $S \subseteq P_c$, define the quantity difference as

$$d(S) = \sum_{i \in S} x_i - N.$$

Here, $d(S) > 0$ represents an excess contract and $d(S) < 0$ represents a shortage. The agent scans all $S \subseteq P_c$ and finds the surplus-side subset S^+ that is closest to the need and the shortage-side subset S^- that is closest to the need. If multiple candidates have the same quantity difference, the agent prefers the higher-price set when it is selling and the lower-price set when it is buying.

3.2. Threshold Acceptance

The threshold-acceptance design is based on CautiousOneshotAgent, the 2024 winner, with the additional use of `sd_ratio`. Let the relative negotiation time be $r \in [0, 1]$, and let the acceptable shortage-side and surplus-side thresholds be $\theta^-(r)$ and $\theta^+(r)$. In the implementation, these are not fixed values; they change as the negotiation progresses. If the shortage-side base value is u^- , the surplus-side base value is u^+ , and the exponent parameter is $\gamma = \text{mismatch_exp}$, then the shortage-side threshold is

$$\theta^-(r) = u^-(1 - r)^\gamma.$$

Since $u^- < 0$, $\theta^-(r)$ approaches 0 as r increases, so the tolerance for shortage shrinks over time.

In the normal curve mode, the surplus-side threshold is given by

$$\theta^+(r) = u^+ r^{\frac{1}{\gamma}}.$$

Therefore, as r increases, $\theta^+(r)$ becomes larger, meaning that excess acceptance becomes easier toward the end of the negotiation. The implementation also has a linear mode, in which case

$$\theta^+(r) = cr.$$

Here, c corresponds to `overmismatch_sell_linear * n_lines / sd_ratio` on the selling side and `overmismatch_buy_linear * n_lines * sd_ratio` on the buying side.

If

$$\theta^-(r) \leq d(S^-) \quad \text{or} \quad d(S^+) \leq \theta^+(r)$$

holds, the agent accepts one of the candidate subsets. In other words, the early phase allows a relatively wide shortage-side margin while strongly suppressing surplus-side acceptance; toward the end of the negotiation, the shortage-side threshold becomes stricter and the surplus-side threshold becomes looser. If both candidates fall within the acceptable range, the agent compares the SCML utility values $U(S)$ according to the current setting and selects the one with higher utility. Only when no sufficiently good accepted set is found at this stage does the agent move to the three-state DP to choose counter offers.

3.3. Delta-Based Trinary Model

The three states in counter proposals are represented, as in first proposals, by A , C , and R . In the implementation, however, C is called `neutral`, meaning that the partner does not accept the agent's counter offer as it is, but instead returns another offer and the negotiation continues.

A major difference between counter proposals and first proposals is that probability estimation is based not on the proposed quantity itself but on the quantity change from the incoming offer. If x_i is the quantity offered by partner i and q_i is the counter quantity returned by the agent, then

$$\delta_i = q_i - x_i.$$

The three-state probabilities are estimated for each partner i , negotiation side s , quantity-difference bucket $b(\delta_i)$, and attempt bucket u_i :

$$(p_i^A, p_i^C, p_i^R) = f(i, s, b(\delta_i), u_i).$$

Smoothing using observation counts and prior weights is performed in the same form as in first proposals. However, in counter proposals, the agent does not use a uniform prior distribution. Instead, it uses pre-learned three-state probabilities depending on the supply-chain level, attempt bucket, and quantity difference. In addition, observations with the same sign of δ are weighted

according to distance, allowing the agent to use nearby experience even when there are few samples with exactly the same quantity difference.

3.4. Continuation-Aware Counter Search

A candidate action g is represented by a set H of incoming offers to accept, a set K of partners to counter, and counter quantities q_i . Offers in H are accepted immediately, while partners in K receive counter offers. Let K^+ be the set of partners receiving a positive counter quantity. The probability of response $y_i \in \{A, C, R\}$ for each partner is approximated by

$$P(y \mid g) = \prod_{i \in K^+} p_i^{y_i}.$$

Partners in state A accept the counter offer, while partners in state R are removed from the negotiation. By contrast, partners in state C remain as active offers in the next round, so the agent does not terminate evaluation immediately but instead evaluates their continuation value.

Let h be the horizon, B be the set of already fixed offers, and O be the current active-offer set. The expected value of candidate action g is

$$E_{h(g)} = \sum_{y \in \{A, C, R\}^m} P(y \mid g) W_{h(y, g)}.$$

If no partner is in state C , $W_{h(y, g)}$ is the value obtained by applying the SCML utility function U to the fixed offer set. If one or more partners are in state C , their offers form the next state O_y , and the value is evaluated recursively with the horizon reduced by one:

$$W_{h(y, g)} = V_{h-1}(B_y, O_y, Q_y).$$

The remaining target quantity is updated using the accepted quantity a_y at that point:

$$Q_y = (Q - a_y) \vee 0.$$

Thus, the evaluation considers not only the probability that the current counter offer will be accepted, but also how much can be recovered in the next round if the partner returns a counter offer.

3.5. DP Replacement Rule

Finally, the action g^* with the maximum expected value is selected from the candidate set G :

$$g^* \in G, E_{h(g^*)} \geq E_{h(g)}, \forall g \in G.$$

However, the DP result is not always used. Let the expected value of the normal counter action be V_{base} , the best expected value found by DP be V_{dp} , and the replacement margin be μ . The DP action replaces the normal action only when

$$V_{\text{dp}} > V_{\text{base}} + \mu.$$

In the submitted version, $h = 3$ and $\mu = 0$, so the DP-based counter proposal is executed only when it is evaluated to have higher expected utility than the normal action.