

SugaiAgent: An agent submitted to the ANAC 2026 SCM league

Toshiki Sugai¹, Takanobu Otsuka²

^{1,2}Nagoya Institute of Technology, Aichi, Japan

¹sugai.toshiki@otsukalab.nitech.ac.jp

²otsuka.takanobu@nitech.ac.jp

June 24, 2026

Abstract

SugaiAgent is a negotiation agent for the standard (Std) track of the SCM league. It handles negotiations with suppliers and consumers concurrently within a single synchronized loop, and is built around three ideas: *trust-weighted quantity allocation* across partners, *time-dependent price concession*, and *greedy acceptance that never exceeds the needed quantity* (over-contracting prevention). Across five tournament runs against three built-in baselines (GreedyStdAgent, SyncRandomStdAgent, and RandomStdAgent), SugaiAgent ranked first in every run, achieving a mean score of 1.055 ± 0.092 (mean $\pm$ SD), beating GreedyStdAgent by about 25% and SyncRandomStdAgent by about 49%. This report describes the design rationale, the implementation of concurrent negotiation and risk management, and the evaluation results.

1 Introduction

In the Std track of the SCM league, an agent acts as a middle-level trader that buys raw materials from suppliers and sells products to consumers at the same time. Profit is maximized by securing the needed quantity, signing contracts at favorable prices, and *avoiding the penalties of disposal (leftover stock) and shortfall (unmet demand)*. A strong agent therefore needs, simultaneously, (i) a pricing strategy that extracts favorable terms from each partner, (ii) control that drives many concurrent negotiations, and (iii) risk management so that signed contracts do not exceed the needed quantity.

SugaiAgent is implemented as a synchronized agent extending **StdSyncAgent**, deciding the responses to all partners together in each negotiation round. At the start of every step (**before_step**) it reads the tradable price range and the needed quantities, and uses them to propose, counter, and accept. The following sections describe the design.

2 The Design of the Agent

The overall flow is shown in Figure 1. In each step, partners are split into a selling side (consumers) and a buying side (suppliers), and each side is processed independently according to its own need (**needed_sales** / **needed_supplies**).

2.1 Negotiation Choices

Normalized price utility. The desirability of a price is normalized to $[0, 1]$ per direction (**_price_utility**). On the selling side a higher price is better, on the buying side a lower price is better; the value is mapped linearly within that day’s price range $[p_{\min}, p_{\max}]$. This lets buying and selling be compared in a single framework.

Opening price. The agent opens from its own favorable end and concedes only slightly (**_opening_price**). The concession width grows with the trust τ as $0.1 + 0.2\tau$, so it makes a small move toward partners with a high record of agreement.

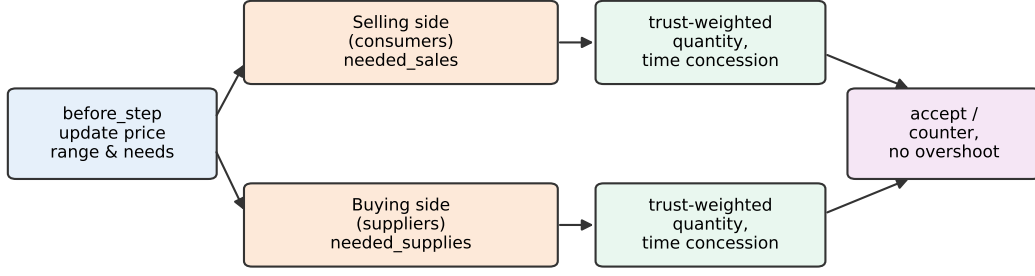


Figure 1: Two-sided synchronized negotiation in a single step. Each side performs trust-weighted quantity allocation and time-dependent price concession, and acceptance is capped so as not to exceed the needed quantity.

Time-dependent concession. The target price moves toward the unfavorable end as relative time $t \in [0, 1]$ advances (`_target_price`, concession factor $0.1 + 0.7t$). The counter price is the convex combination of the partner’s offered price and the agent’s own target,

$$p = \alpha p_{\text{offer}} + (1 - \alpha) p_{\text{target}}, \quad \alpha = \begin{cases} 0.8 & (t > 0.8) \\ 0.3 + 0.5\tau & (\text{otherwise}) \end{cases}$$

(`_counter_price`). The later the round or the more trusted the partner, the more weight is put on the partner’s offered price, converging to agreement faster.

Acceptance decision. The absolute criterion for acceptance is a time-relaxed threshold $\theta = 0.6(1 - t)$ on the normalized price utility. Early on, only high-profit offers are accepted; later the threshold is lowered to prioritize securing the needed quantity. The ranking itself uses the utility function `ufun`, and offers whose price utility exceeds the threshold are selected from the top down.

2.2 Concurrent Negotiation

Under the `StdSyncAgent` framework, `first_proposals` and `counter_all` return the responses to all partners simultaneously. `SugaiAgent` divides partners into the selling and buying sides and processes each independently in `_counter_side`.

First proposals. The needed quantity is allocated across partners weighted by trust. The proposed quantity for partner p blends the trust-based share $\text{need} \cdot \tau_p / \sum_q \tau_q$ with the historical average contract quantity `_avg_quantity` in equal parts, $q_p = \text{round}(0.5 \text{ need } w_p + 0.5 \bar{q}_p)$, capped by the need. This assigns more to partners that tend to agree and less to new partners.

Counters and acceptance. `counter_all` ranks the incoming offers by utility and greedily accepts those above the threshold θ *without exceeding the need* ($\text{secured} + q \leq \text{need}$). For the remaining need, it counters by requesting the smaller of the remaining quantity and the partner’s offered quantity, while respecting the partner’s desired delivery time (`TIME`). It always returns a response to every partner and returns `None` (end negotiation) on the unnecessary side to avoid missing responses.

2.3 Risk Management

Over-contracting prevention. This is the most important design point of the agent. On a successful agreement (`on_negotiation_success`) the need on that side is decremented by the contracted quantity, and negotiations on a side whose need reaches zero or below are ended immediately. In addition, the acceptance stage caps the cumulative quantity at the need, and the counter quantity is bounded by the remaining need, so leftover stock and over-buying are structurally unlikely.

Trust estimation. The agreement probability per partner is estimated with Laplace smoothing, $\tau_p = (s_p + 1)/(t_p + 2)$ (with s successes and t trials). An unknown partner starts from 0.5, and division by zero is avoided.

Early termination of unpromising negotiations. A partner with more than 5 trials, trust below 0.2, and a negotiation already in its second half ($t > 0.5$) is dropped, redirecting the limited negotiation steps toward more promising partners.

Robustness. The price range and needs are refreshed every step in `before_step`, and the retrieval of price and delivery time falls back gracefully on exceptions (`_safe_price`, `_delivery_time`), so the agent runs without crashing even when issues are missing.

3 Evaluation

Setup. Five independent tournaments were run with `n_steps= 50`, `n_configs= 5`, `n_runs_per_world= 1` (60 worlds per run) against three built-in baselines: GreedyStdAgent, SyncRandomStdAgent, and RandomStdAgent. All runs use the same four competitors to make variance across runs directly comparable.

Normalized score. The score reported by `anac2024_std` is each agent’s total profit divided by a world-level reference value (the maximum profit achievable under the world configuration). A score of 1.0 means the agent’s profit equalled this reference; values above 1.0 arise when the agent negotiates more favorably than the conservative reference assumes (e.g. securing above-reference prices or quantities), while values below 0 indicate net losses from disposal or shortfall penalties.

The score per run is shown in Table 1 and Figure 2.

Table 1: Normalized score per run (60 worlds each), mean, and standard deviation (SD) across five runs.

Agent	Run1	Run2	Run3	Run4	Run5	Mean	SD
SugaiAgent	1.023	1.204	1.079	0.991	0.976	1.055	0.092
GreedyStdAgent	0.936	0.832	0.875	0.775	0.814	0.846	0.062
SyncRandomStdAgent	0.628	0.626	0.749	0.957	0.590	0.710	0.151
RandomStdAgent	−0.053	−0.631	−0.092	−0.285	−0.269	−0.266	0.229

SugaiAgent ranked **first in all five runs**, with a mean score of 1.055 ± 0.092 (mean $\pm$ SD across five runs). This is about **25%** above GreedyStdAgent (0.846 ± 0.062) and about **49%** above SyncRandomStdAgent (0.710 ± 0.151). RandomStdAgent scored negatively on average (-0.266 , SD 0.229), confirming that uncontrolled over-contracting and shortfalls are severely penalized. In Run4 SyncRandomStdAgent temporarily rose above GreedyStdAgent, reflecting the high run-to-run variance of the random baseline (SD 0.151 vs. 0.062 for Greedy); SugaiAgent’s comparatively modest SD (0.092) suggests that the over-contracting prevention and need-tracking provide robustness across world configurations.

3.1 Ablation Study

To quantify the sensitivity of the three key hyper-parameters, we ran one-at-a-time sweeps: each parameter is varied over a grid while the other two are held at their defaults ($\beta = 0.5$, $\theta_0 = 0.6$, $c = 0.7$). Each grid point is evaluated in a single tournament (`n_configs= 5`, `n_runs_per_world= 1`, `n_steps= 50`, ~ 60 worlds) against the same three baselines. Results are shown in Table 2 and Figure 3.

Blend weight β . The score peaks at $\beta=0.75$ (1.092) and drops at both extremes. A pure historical-average allocation ($\beta=0$) and a pure trust-weighted one ($\beta=1$) both score around 0.94–0.97, while the blend at 0.75 achieves the best of both: trust directs more proposals toward reliable partners, and

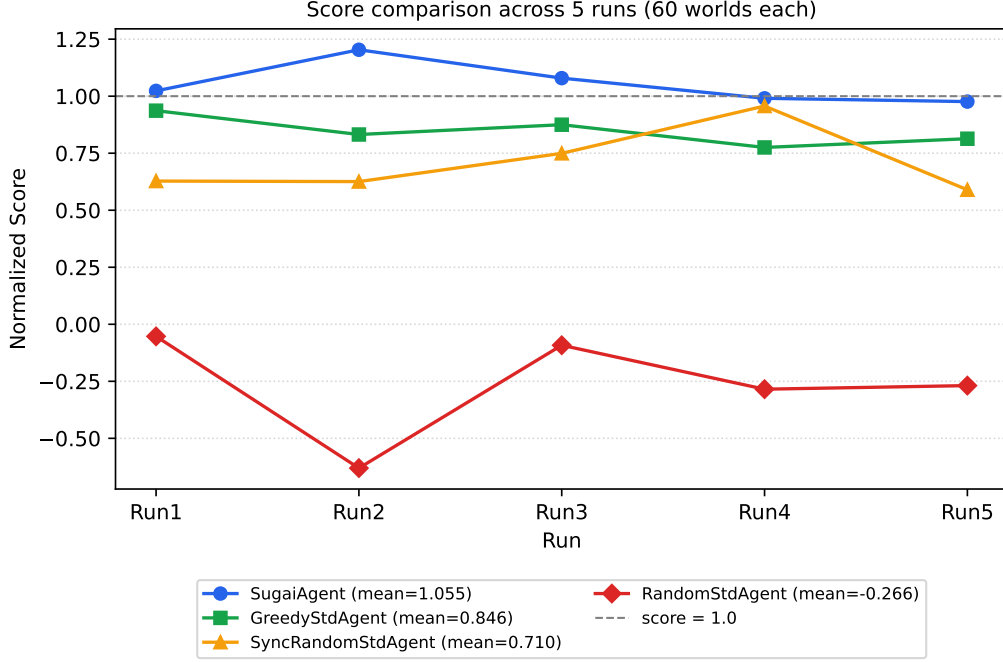


Figure 2: Score comparison per run (dashed line at score 1.0). SugaiAgent is ranked first in all five runs; error bars show the inter-run variability visible in the table.

Table 2: Ablation scores for each parameter. **Bold** marks the default value; underline marks the best value.

(a) Blend weight β ($\theta_0=0.6$, $c=0.7$ fixed)					
β	0.00	0.25	0.50	<u>0.75</u>	1.00
Score	0.936	0.914	0.967	<u>1.092</u>	0.965
(b) Threshold coeff θ_0 ($\beta=0.5$, $c=0.7$ fixed)					
θ_0	0.30	0.50	0.60	<u>0.70</u>	0.90
Score	0.925	0.943	0.970	<u>1.143</u>	1.040
(c) Concession coeff c ($\beta=0.5$, $\theta_0=0.6$ fixed)					
c	0.30	0.50	0.70	<u>0.90</u>	
Score	0.825	0.967	0.930	<u>0.976</u>	

the historical average prevents the proposal from collapsing to zero for new partners. The default $\beta=0.5$ is reasonable but not optimal; $\beta=0.75$ would be a better choice.

Threshold coeff θ_0 . This is the most sensitive parameter. The score rises monotonically from $\theta_0=0.3$ (0.925) to $\theta_0=0.7$ (1.143), then falls at $\theta_0=0.9$ (1.040). A stricter early threshold (larger θ_0) prevents accepting low-quality offers while the deadline is still far away, yielding higher margins; however, if θ_0 is too large (0.9), the agent rejects too many offers and misses quantity targets. The default $\theta_0=0.6$ is conservative; setting $\theta_0=0.7$ improves the mean score by about 18% relative to the default.

Concession coeff c . The score is relatively flat across the grid (0.825–0.976), making c the least sensitive parameter. The numerical maximum is at $c=0.9$ (0.976), but the gap over $c=0.5$ (0.967) is only 0.009—within single-run noise—so no specific value is robustly optimal. Very slow concession ($c=0.3$, score 0.825) is the only clearly suboptimal setting: the target price stays near the agent’s favorable extreme even at the deadline, leading to few agreements. Once the schedule is steep enough

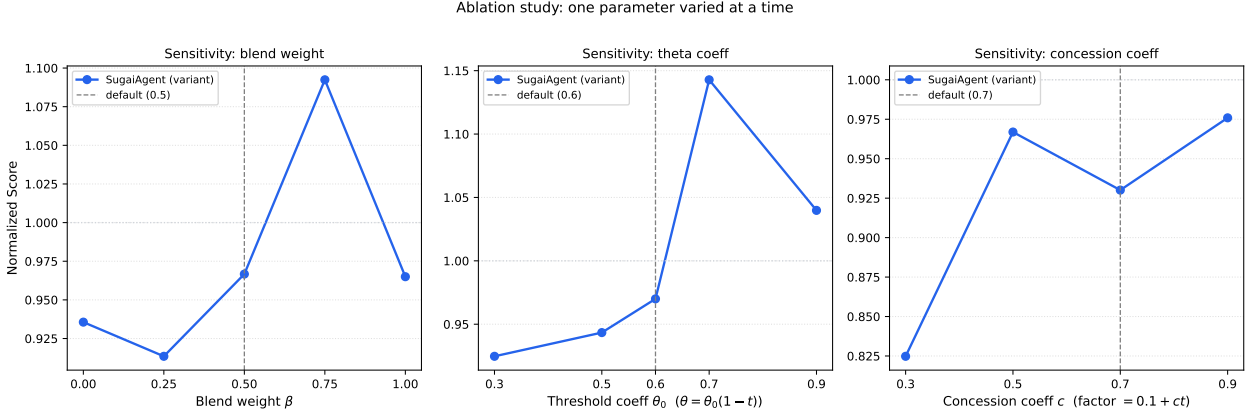


Figure 3: Sensitivity of SugaiAgent’s score to each hyper-parameter (dashed vertical line = default value). Left: blend weight β ; centre: threshold coeff θ_0 ; right: concession coeff c .

to reach agreement before the deadline ($c \geq 0.5$), the exact slope is secondary.

4 Lessons and Suggestions

The evaluation confirms that simple rules—acceptance that never exceeds the need and trust-weighted quantity allocation—yield a stable advantage over the naive Greedy strategy. There is, however, room for improvement. First, trust is defined solely by agreement probability, so the *quality of the agreed price* is not reflected; evaluating partners by the product of price utility and agreement probability (expected utility) could reduce the share of low-margin agreements. Second, the need relies on the per-day value returned by the AWI and does not *look ahead* to future-step demand or to inventory and exogenous-contract dynamics. Third, fixing the delivery time to its earliest value underuses the available scheduling freedom.

As suggestions: (i) learn each partner’s concession curve to adapt the counter weight α ; (ii) dynamically adjust the acceptance threshold θ to the competitive density of the market; and (iii) introduce a quantity plan that anticipates supply and demand several steps ahead. These three directions appear most promising.

Conclusions

SugaiAgent is a negotiation agent that combines trust-weighted quantity allocation, time-dependent price concession, and greedy acceptance that never exceeds the need, on top of synchronized two-sided concurrent negotiation. By structurally preventing over-contracting, it suppresses disposal/shortfall penalties and consistently beat GreedyStdAgent and SyncRandomStdAgent over five tournaments (by about 25% and 49% in mean score, respectively; SD 0.092). An ablation study confirms that the acceptance threshold (θ_0) is the most sensitive parameter: setting $\theta_0=0.7$ instead of the default 0.6 raises the mean score by $\sim 18\%$, and shifting the blend weight to $\beta=0.75$ offers a further gain. The concession rate c is insensitive once $c \geq 0.5$; only very slow concession ($c=0.3$) clearly hurts. Future work aims to further improve profit through expected-utility-based partner evaluation, look-ahead demand planning, and incorporation of the ablation-identified settings ($\theta_0=0.7$, $\beta=0.75$) into the final agent.