

Design and Implementation of the heystd Agent for the SCML Standard Track

kazuma mochizuki

June 16, 2026

Abstract

The Supply Chain Management League (SCML) presents a highly competitive and computationally constrained environment for autonomous negotiating agents. This report details the architecture and underlying algorithms of the *Std* agent, a highly optimized and robust synchronous agent specifically designed for the SCML Standard Track. To prevent critical timeout errors commonly associated with large-scale multi-lateral negotiations, *Std* completely eliminates traditional combinatorial powerset searches, replacing them with a highly efficient greedy heuristic algorithm. The agent progressively evaluates and accepts offers based on profitability, prioritizing the lowest unit prices from suppliers and the highest from consumers. Furthermore, *Std* implements a time-dependent dynamic concession strategy, flexibly adjusting its pricing expectations based on the simulation's progress to increase agreement rates as deadlines approach. By securely managing inventory risks through the random distribution of urgent daily needs among major partners and proactively managing future offers up to three steps ahead, the agent aims for an aggressive 90% target productivity. This report explores how *Std* effectively maximizes total financial scores while strictly honoring systemic computational boundaries.

1 Introduction

The Supply Chain Management League (SCML) is an international competition designed to evaluate the performance of autonomous agents in managing dynamic, multi-tier supply chains. In the Standard Track, agents act as factory managers who must simultaneously negotiate with suppliers for input materials and with consumers to sell their manufactured products. A primary challenge in this environment is the strict computational time limit imposed on negotiations. Agents that employ complex, exhaustive search algorithms often fail due to timeout errors, resulting in lost negotiation opportunities and severe financial penalties.

The *Std* agent was conceptualized and developed to address these specific challenges. Rather than striving for absolute theoretical optimality through exhaustive state-space exploration, *Std* focuses on computational robustness, practical efficiency, and high throughput. The core philosophy of *Std* is to achieve an aggressive target productivity of 90% while strictly adhering to the computational constraints of the SCML simulation environment.

2 Algorithmic Foundation: Eradicating Combinatorial Explosions

One of the most significant pitfalls for agents in the SCML environment is the evaluation of concurrent offers. Traditionally, agents might attempt to evaluate all possible subsets of received offers (a powerset search) to find the combination that maximizes immediate profit while satisfying factory constraints. If an agent receives N offers, the powerset evaluation requires $O(2^N)$ computational steps. In scenarios involving dozens of negotiating partners, this exponential time complexity guarantees a system timeout.

The *Std* agent completely bypasses this combinatorial explosion by implementing a **Greedy Heuristic Algorithm** for offer evaluation.

2.1 Greedy Offer Evaluation Strategy

When the `counter_all` method is invoked, *Std* processes incoming offers independently for its supply and consumption channels. The algorithm operates as follows:

1. **Filtering:** Offers that do not match the current simulation step or fall outside valid price bounds are immediately discarded or deferred.
2. **Sorting by Profitability:** The agent sorts the remaining valid offers based on the unit price. When buying from suppliers, the list is sorted in ascending order (prioritizing the lowest price). When selling to consumers, it is sorted in descending order (prioritizing the highest price).
3. **Sequential Acceptance:** The agent iterates through the sorted list, accumulating quantities. It accepts offers sequentially until its calculated daily needs (based on a 90% target productivity) are fulfilled. A buffer margin of 10% of the total factory lines is added to prevent minor shortfalls.

This greedy approach reduces the complexity of offer evaluation from $O(2^N)$ to $O(N \log N)$ due to the sorting step, ensuring immediate execution regardless of the number of active negotiation partners.

3 Pricing and Dynamic Concession Strategy

Static pricing strategies are generally ineffective in dynamic supply chains. Agents that refuse to lower their expectations often fail to secure contracts, leading to idle factories and inventory bloat. Conversely, agents that concede too quickly leave potential profits on the table. *Std* employs a **Time-Dependent Dynamic Concession Strategy**.

3.1 Concession Function Formulation

The pricing logic implemented in the `price()` method computes the desired unit price dynamically based on the current progress of the simulation. Let P_{min} and P_{max} represent the minimum and maximum valid values for the unit price issue in a given negotiation. Let $S_{current}$ be the current simulation step, and S_{total} be the total number of simulation steps.

The progress ratio is defined as:

$$\text{Progress} = \frac{S_{\text{current}}}{\max(1, S_{\text{total}})} \quad (1)$$

The target price for consumer negotiations (selling) is calculated as:

$$Price_{\text{consumer}} = P_{\text{max}} - (P_{\text{max}} - P_{\text{min}}) \times (0.1 + 0.6 \times \text{Progress}) \quad (2)$$

The target price for supplier negotiations (buying) is calculated as:

$$Price_{\text{supplier}} = P_{\text{min}} + (P_{\text{max}} - P_{\text{min}}) \times (0.1 + 0.6 \times \text{Progress}) \quad (3)$$

3.2 Strategic Implications

At the beginning of the simulation ($\text{Progress} \approx 0$), the agent is highly aggressive. It demands prices near the absolute maximum from consumers and offers near the absolute minimum to suppliers, conceding only 10% of the negotiation range. As the simulation approaches its deadline ($\text{Progress} \approx 1$), the agent gradually softens its stance, conceding up to 70% of the range to ensure that final contracts are secured. This flexibility increases the overall agreement rate and keeps the supply chain flowing steadily.

4 Inventory and Risk Management

Maintaining an optimal inventory level is crucial in SCML. Overstocking raw materials ties up capital and incurs storage costs, while understocking leads to idle production lines. *Std* tackles this via distributed daily needs and forward contract planning.

4.1 Distribution of Urgent Daily Needs

When the agent detects a shortfall for the current day's production, it does not rely on a single supplier to fulfill the entire deficit. Relying on a single partner poses a severe risk; if that partner fails to agree or deliver, the factory halts.

Instead, *Std* utilizes the `distribute_todays_supplie_consume_needs` method. This algorithm shuffles the list of available partners and selects a random subset, defined by the `ptoday` parameter (default set to 0.75, or 75% of partners). The required quantity is then mathematically distributed across these bins. This multi-sourcing strategy acts as a risk-mitigation mechanism, ensuring that partial failures in negotiation do not result in total production failure.

4.2 Future Offer Generation and Acceptance

To further stabilize its supply chain, *Std* proactively engages in future planning. The agent calculates its expected production needs up to three steps ahead. The target partners are divided into three tiers:

- **Tier 1 (First 50%):** Targeted for offers 1 step ahead.
- **Tier 2 (Next 30%):** Targeted for offers 2 steps ahead.

- **Tier 3 (Remaining 20%):** Targeted for offers 3 steps ahead.

By distributing future requests across different time horizons and different partner groups (`future_supplie_offer` and `future_consume_offer`), *Std* secures its necessary inputs and outputs in advance. When evaluating incoming proposals (`counter_all`), if an offer is for a future step and fits within the projected needs, the agent immediately issues an `ACCEPT_OFFER` response, securing the contract before immediate deadlines create pressure.

5 Conclusion