

Proposal and Analysis of GS3

Rintaro Nakayama, Shoma Mizuno
Nagoya Institute of Technology

June 24, 2026

Abstract

This report describes the design and experimental evaluation of GS3, an autonomous negotiation agent for the SCML Standard Track. GS3 employs Q-learning to drive its process. The objective of GS3 is to build a system that avoids the explosion of the state space caused by vast combinations of contracts, thereby enabling decision-making within limited time constraints. Since directly learning whether to accept or reject individual contracts is difficult due to the vastness of the search space, we adopted an approach that separates high-level strategy determination from specific contract selection. The proposed method is supported by three key components: a strategy selection layer that determines optimal evaluation weights using linear function approximation Q-learning, a contract approximation optimization solver that takes these weights as parameters to derive the optimal set from simultaneous offers, and risk-averse production control that preemptively blocks losses during extreme deflationary periods. Experimental results demonstrate that our approach outperforms last year’s tournament-winning agent, proving its effectiveness.

1 Introduction

GS3 agent proposed in this study addresses the limitations of conventional tabular Q-learning—namely, its low generalization performance to unseen states and the explosion of the state space—by introducing linear function approximation to the state representation. The strength of GS3 lies in its decoupled architecture that combines high-level strategy determination through reinforcement learning with localized contract optimization through dynamic programming. Specifically, instead of directly accepting or rejecting individual contracts, the reinforcement learning layer selects a strategic option based on market conditions, such as profit-oriented, demand-fulfillment-oriented, or trust-oriented strategies. On the other hand, the actual contract selection is formulated as a knapsack problem, which is optimized via dynamic programming using evaluation function weights and penalty coefficients corresponding to the selected high-level strategy. The remainder of this report is organized as follows: Section 2 describes the detailed specifications of Q-learning in the proposed GS3 agent, including the design of the action and state spaces, the formulation using linear function approximation, and the reward function. Section 3 elaborates on the knapsack optimization via dynamic programming into which the outputs of Q-learning are injected, along with its mapping to the mathematical model. Section 4 presents the asset transitions during the pre-offline learning phase and the results of tournament evaluation experiments in an online environment with competing agents, validating the effectiveness of the proposed method. Finally, Section 5 discusses future prospects, and Section 6 concludes this report.

2 Detailed Specifications of Q-learning

2.1 Action Space Definition

Let $\mathcal{A}$ be the set of the action space in reinforcement learning. We mathematically define it as five strategic options as follows:

$$\mathcal{A} = \{\text{PROFIT}, \text{BALANCE}, \text{NEED}, \text{TRUST}, \text{REJECT}\}$$

- PROFIT (Profit-oriented strategy): Maximizes profit.
- BALANCE (Harmonized strategy): Optimizes the balance among profit, partner trust, and required volume fulfillment.
- NEED (Volume-fulfillment strategy): Prioritizes securing the required production volume above all else.

- **TRUST** (Relationship-oriented strategy): Favors “excellent partners” who have a rich history of transaction success rates and quantities.
- **REJECT** (All-rejection strategy): Rejects all offers in the current step.

2.2 State Space Design

The state space of GS3 is configured as a 13-dimensional feature vector $\mathbf{x} \in \mathbb{R}^{13}$. Each feature complies with SCML domain terminology and environment variables, consisting of the following four categories that represent the agent’s internal environment and external market signals:

1. **Internal Physical Constraints** ($x_1 \sim x_3$): Current input inventory level, unfulfilled required volume, and remaining simulation steps.
2. **Micro-transaction Environment** ($x_4 \sim x_8$): Advantageousness of the presented offer price, trustworthiness of the negotiation partner based on historical performance, and an internal control flag governing the agent’s trading stance. This flag binary-represents whether the agent is in an “aggressive mode” to accept offers more actively when the risk of shortfalls (and accompanying penalties) is high, or a “conservative mode” to tighten selection to minimize losses during overstocking or deflationary periods.
3. **Macro-market Demand and Supply Signals** ($x_9 \sim x_{12}$): Real-time supply-demand tightness and price trends in both the raw material and product markets, extracted from external market reports.
4. **Bias Term** (x_{13}): A fixed value of 1.0 to represent the intercept in the mathematical formula.

2.3 Formulation of Q-learning Using Linear Function Approximation

GS3 calculates the Q-value for a given state $\mathbf{x}$ and a selected action $a \in \mathcal{A}$ through a linear combination with a 13-dimensional weight vector $\mathbf{w}_a$ prepared individually for each action:

$$Q(\mathbf{x}, a) = \mathbf{w}_a \cdot \mathbf{x} = \sum_{i=1}^{13} w_{a,i} x_i \quad (1)$$

This approximation allows the agent to estimate Q-values even for unseen states that have similar features to learned states, thereby mitigating the state explosion problem inherent in tabular Q-learning.

2.4 Reward Function Design and the Substance of Immediate Rewards

As the substance of the reward, GS3 defines a unique step immediate reward function. At the end of each negotiation step and immediately after contracts are finalized, this function evaluates the quality of the contract set selected by dynamic programming against the physical constraints of the factory on the spot. Specifically, it configures the reward using price advantage P , partner trust T , and required volume fulfillment rate F as positive evaluation terms, and deviation rate from the preferred delivery time D , over-contracting rate O , and under-contracting rate U as negative evaluation terms:

$$\text{reward} = w_P P + w_T T + w_F F - (w_D D + w_O O + w_U U) \quad (2)$$

where $w_P, w_T, w_F, w_D, w_O, w_U$ are the weight coefficients for each evaluation term. This reward is dynamically computed within the agent at the end of each simulation step and immediately passed to the Q-value update step. Let $\mathcal{S}$ be the set of finalized contracts selected by dynamic programming at the current step. For each contract $i \in \mathcal{S}$, let q_i be the contracted volume, a_i be the normalized price advantage of the contract, and s_i be the trust score of the partner. The total contracted volume is denoted by

$$Q = \sum_{i \in \mathcal{S}} q_i \quad (3)$$

Let N be the required factory volume ($N \geq 0$). The raw metrics used in the reward calculation are defined as follows:

$$\begin{aligned} P &= \frac{\sum_{i \in \mathcal{S}} q_i a_i}{\max(1, Q)} & T &= \frac{\sum_{i \in \mathcal{S}} s_i}{\max(1, |\mathcal{S}|)} & F &= \frac{\min(Q, N)}{\max(1, N)} \\ D &= \frac{|Q - N|}{\max(1, N)} & O &= \frac{\max(0, Q - N)}{\max(1, N)} & U &= \frac{\max(0, N - Q)}{\max(1, N)} \end{aligned} \quad (4)$$

The normalized price advantage a_i is computed from the negotiation price range. For selling contracts,

$$a_i = \frac{p_i - p_i^{\min}}{p_i^{\max} - p_i^{\min}} \quad (5)$$

while for buying contracts,

$$a_i = \frac{p_i^{\max} - p_i}{p_i^{\max} - p_i^{\min}} \quad (6)$$

Therefore, larger values of a_i always indicate a more favorable price for the agent regardless of whether it acts as a buyer or a seller.

Finally, the reward value is normalized before being used for Q-learning updates:

$$\tanh\left(\frac{\text{reward}}{3}\right) \quad (7)$$

2.5 Weight Vector Update

During the simulation, the weight vector is updated at the end of each step based on temporal difference learning using the normalized immediate reward r calculated at the timing mentioned above:

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha \delta \mathbf{x} \quad (8)$$

where α is the learning rate and δ is the temporal difference error, which is defined by the following equation:

$$\delta = \left(r + \gamma \max_{a' \in \mathcal{A}} Q(\mathbf{x}', a') \right) - Q(\mathbf{x}, a) \quad (9)$$

Here, γ is the discount rate for considering future rewards.

3 Knapsack Optimization via Dynamic Programming Injected with Q-learning Outputs

The core of the proposed system is to dynamically map high-level decisions made by reinforcement learning into a contract knapsack problem solved via dynamic programming. Let $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$ be the set of simultaneous offers received from negotiation partners. Each offer o_j is characterized by its quantity q_j , normalized price advantage $\text{Price}(o_j)$, and partner trust score $\text{Trust}(o_j)$. Given W as the maximum total quantity acceptable by the factory, the knapsack problem to be solved is formulated as follows:

$$\text{Maximize} \quad \sum_{j=1}^n \mathcal{F}(o_j, a^*) z_j \quad (10)$$

$$\text{subject to} \quad \sum_{j=1}^n q_j z_j \leq W, \quad z_j \in \{0, 1\} \quad (11)$$

where z_j is a binary selection variable indicating whether to accept (1) or reject (0) the offer o_j . The primary novelty of this proposal is that the function $\mathcal{F}(o_j, a^*)$, which determines the final evaluation value of each offer, is dynamically adjusted according to the current optimal strategy $a^* = \arg \max_a Q(\mathbf{x}, a)$ selected by Q-learning. Specifically, using the meta-parameters shown in Table 1 (price weight β_{price} , trust weight β_{trust} , delivery penalty coefficient γ_{time} , over-contracting penalty coefficient γ_{over} , and under-contracting penalty coefficient γ_{under}), the evaluation function $\mathcal{F}(o_j, a^*)$ is constructed as follows:

$$\mathcal{F}(o_j, a^*) = \beta_{\text{price}}(a^*) \text{Price}(o_j) + \beta_{\text{trust}}(a^*) \text{Trust}(o_j) - \text{Penalty}(o_j, a^*) \quad (12)$$

Here, the weighted sum in the first and second terms on the right-hand side corresponds to the base score v_j of the single offer, and the deduction term $\text{Penalty}(o_j, a^*)$ is computed according to the selected strategy a^* as follows:

$$\text{Penalty}(o_j, a^*) = \gamma_{\text{time}}(a^*) \Delta t_j + \gamma_{\text{over}}(a^*) \max(0, q_j - V_{\text{allow}}) + \gamma_{\text{under}}(a^*) \max(0, V_{\text{need}} - q_j) \quad (13)$$

where Δt_j represents the delivery deviation of the offer from the ideal factory schedule, V_{allow} is the capacity capable of avoiding over-contracting penalties, and V_{need} is the minimum unfulfilled production obligation required to avoid shortfall penalties at that step.

Table 1: Correspondence between the selected strategy $a \in \mathcal{A}$ and DP evaluation function weights/penalties

Selected Strategy a	Price Weight (β_{price})	Trust Weight (β_{trust})	Delivery Penalty (γ_{time})	Over-contract Penalty (γ_{over})	Under-contract Penalty (γ_{under})
PROFIT	3.20	0.80	1.50	3.00	1.30
BALANCE	2.40	1.30	1.90	2.20	1.75
NEED	2.10	1.00	1.75	2.00	2.55
TRUST	1.50	3.00	1.90	2.70	2.00
REJECT	0.00	0.00	0.00	0.00	0.00

Based on this mathematical model, dynamic programming is utilized to select the optimal combination of offers z_j^* that maximizes the objective function. Furthermore, when REJECT is selected as the optimal strategy, or when the market is in an extreme deflationary state, production control that temporarily halts factory operations is linked, preventing unnecessary raw material consumption and excess inventory penalties.

4 Experimental Results: Comparative Analysis of Asset Changes and Learning Progress

In this section, we conduct a two-stage experimental evaluation to verify the effectiveness of the proposed GS3 agent. First, in Sections 4.1 and 4.2, we perform reinforcement learning in a multi-agent environment to analyze the learning progress, the agent’s asset change (**balance delta**), and the average reward (**mean reward**). Subsequently, Section 4.3 presents the results of tournament evaluation experiments where the agent—with its fully mature, trained weight vectors fixed—competes simultaneously against various rival agents, including last year’s winning agent. Epsilon is fixed at 0.01 in the late stages of training. Table 2 shows the Q-learning and environmental hyperparameter settings used in this experiment.

Table 2: Hyperparameter configurations for Q-learning and the experimental environment

Parameter	Value
Learning Rate (α)	0.05 $\sim$ 0.10 (depends on the number of learned states)
Discount Rate (γ)	0.85
Exploration Rate (ϵ)	0.01 $\sim$ 0.10 (depends on the number of learned states)
State Feature Dimensions	13
Total Steps per World	50
Number of Configs	20

4.1 The Early Learning Stage

Figure 1 illustrates the **mean reward** and **balance delta** of GS3 during the initial learning sets. Because the weight vector $\mathbf{w}_a$ is immature, a phenomenon is observed where the **mean reward** (blue line) drops significantly toward the negative side from around steps 10 to 25. This is likely due to the selection of unfavorable contract strategies in an unknown environment before an optimal strategy could be established.

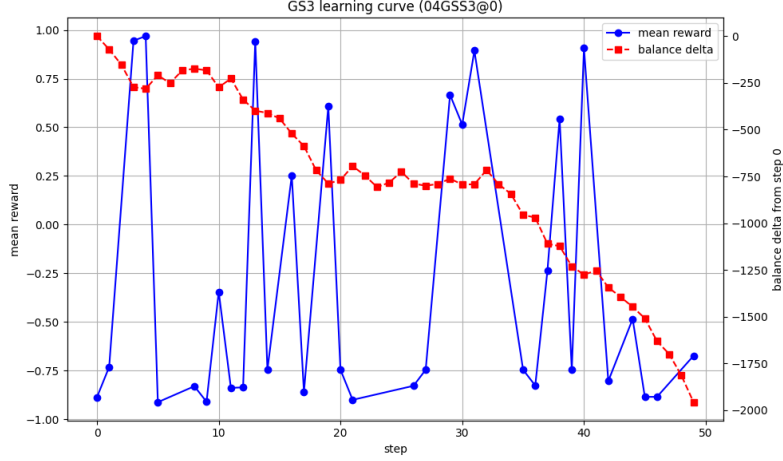


Figure 1: The early learning stage

4.2 The Late Learning Stage

Figure 2 shows the **balance delta** at the final stage after a sufficient number of training sets, where the weight vector has become mature and optimized. Compared to the early stage (Figure 1), the result has clearly improved.

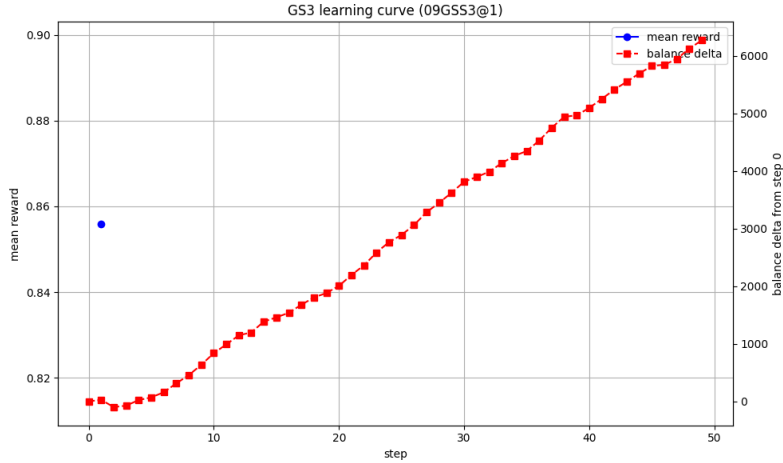


Figure 2: The late learning stage

The growth of the **balance delta** increases linearly from immediately after the initial steps until the end of the simulation.

4.3 Tournament Results with Multiple Agents

To verify the superiority of the proposed method, we conducted tournament evaluations in a multi-agent simulation environment with other agents, including last year's top-performing agents. As the experimental setup, the tournament was run 5 times with 20 configurations, executing 5 simulations per world (50 steps per world). Comparative experiments were conducted based on the Q-values obtained from these total 5 sets of trials. The competing agents are configured as follows:

- **GS3**: The proposed method in this study.
- **GS2**: Conventional learning strategy model (with in-tournament learning, without linear approximation).
- **GS1**: Complete offline strategy model (fully offline during the tournament, without linear approximation).

- **AS0**: Last year’s tournament winner.
- **PriceTrendStdAgent**: Participant in last year’s tournament.
- **XenoSotaAgent**: Participant in last year’s tournament.
- **UltraSuperMiracleSoraFinalAgentZ**: Participant in last year’s tournament.
- **KATSUDONAgent**: Participant in last year’s tournament.

Table 3 summarizes the mean, variance, maximum, and minimum scores of each agent obtained from the simulation experiments.

Table 3: Statistical comparison of scores among agents in the tournament evaluation

Agent Name	Mean	Variance	Maximum	Minimum
GS3 (Proposed)	1.0264	0.1090	1.8806	-0.3158
GS2	1.0137	0.0953	1.8812	-0.2586
GS1	1.0155	0.0949	1.6295	-0.3021
AS0 (Last Year’s Winner)	1.0148	0.0934	1.8792	-0.2784
PriceTrendStdAgent	0.9955	0.0809	1.8031	-0.4029
XenoSotaAgent	1.0015	0.1163	1.7934	-0.3927
UltraSuperMiracleSoraFinalAgentZ	0.9881	0.0990	1.7892	-0.5818
KATSUDONAgent	0.7265	0.0636	1.5020	-0.6428

The experimental results reveal that the proposed **GS3** achieved the highest mean score (1.0264), outperforming last year’s winning agent **AS0**, the older model **GS2** without linear approximation, and all other competing agents. This demonstrates that GS3 functions effectively even in tournament environments populated by diverse opponents.

5 Remain issues

The proposed system architecture has demonstrated steady improvement in balance delta as learning progresses. For further performance enhancement, the following approaches can be considered:

1. **Validation of Weight Vectors Using Multiple Regression Analysis**: By utilizing Newey-West standard errors to correct for time-series autocorrelation in the gathered data logs, t -tests and F -tests can be performed to eliminate statistically insignificant features, thereby achieving a more lightweight model.
2. **Partial Introduction of Non-linearity**: Cross-terms combining different state features can be manually appended to the feature vector. This expands the expressive power to capture complex synergetic effects while remaining within the framework of a linear model.

6 Conclusion

In this report, we presented the design and experimental results of GS3, a new agent that integrates linear function approximation Q-learning with dynamic programming. Reinforcement learning manages high-level strategy determination, while the DP solver executes strict contract combination optimization. By introducing mathematical formulations tailored to each high-level strategy, the agent achieved a consistent upward trend in balance delta toward the late stages of learning, ultimately surpassing last year’s winning agent in terms of mean score.