AnalysisAgent: An agent submitted to the ANAC 2025 SCM league

Eito Sugita, Reita Kaneko
Nagoya Institute of Technology
e.sugita.348@stn.nitech.ac.jp, cmt14035@ict.nitech.ac.jp

June 14, 2025

Abstract

This report describes the AnalysisAgent for the SCML OneShot Track of ANAC2025. AnalysisAgent is an agent based on the SimpleAgent that is developed to address the negotiation problems faced by SimpleAgent in a world populated by many asynchronous agents.

1 Introduction

AnalysisAgent is a negotiation agent that is enhanced based on SimpleAgent. In SCML OneShot Track, agents can be broadly classified into two types: those that negotiate sequentially one counterpart at a time like SimpleAgent, and those that evaluate and negotiate multiple interactions synchronously like OneShotSyncAgent. In the 2024 tournament, the top 5 agents were all based on the OneShotSyncAgent framework, while agents based on the SimpleAgent tended to achieve lower scores.

As a preliminary experiment, we had SimpleAgent negotiate with multiple OneShotSyncAgent - based agents within the same world, and observed a similar trend. There are two possible reasons why SimpleAgent achieves lower scores than agents based on the OneShotSyncAgent framework. The first reason is SimpleAgent's proposal strategy. In this approach, SimpleAgent directly offers the required negotiation quantity to all agents. On the other hand, OneShotSyncAgent compares the combinations of all offers made by negotiating agents and accepts the one that most closely meets the required negotiation quantity. Therefore, depending on the combination of agents involved in the negotiation, the probability of SimpleAgent's offer being accepted by OneShotSyncAgent decreases as the required negotiation quantity of SimpleAgent increases.

The second reason is SimpleAgent's acceptance strategy, which disregards ongoing negotiation offers. SimpleAgent accepts every proposal as long as the quantity of offers below its required negotiation quantity, without considering the ongoing negotiation success. Therefore, if a negotiation turns out to be

successful, the cumulative transaction quantity up to that point might exceed the required negotiation quantity. For example, when negotiating with two agents, if SimpleAgent accepts an offer from one agent while offering the full required negotiation quantity to the other, the final transaction quantity can exceed its actual requirement once the SimpleAgent's offer is accepted. Thus, AnalysisAgent investigates whether incorporating the following improvements to SimpleAgent can bring its performance closer to that of OneShotSyncAgent-based agents.

1. Distribution Strategy: By distributing the required negotiation quantity among the negotiating agents, excessive offers are suppressed.

2. Acceptance Strategy: Offers are evaluated for acceptance by considering the expected successful quantity relative to the trading quantity during negotiations.

2 Strategy design

AnalysisAgent negotiates with a single counterpart agent i, using its negotiation history to reach an agreement on both the quantity q_i and the unit price c_i. The negotiation process consists of two strategies: the propose strategy, in which it presents the offer (q_i, c_i) to the counterpart, and the response strategy, in which it decides whether to accept or reject an offer from the counterpart. The following sections provide a detailed explanation of each strategy.

2.1 The propose strategy

AnalysisAgent determines its offers based on its past negotiation history. In doing so, it adjusts the distribution strategy according to the volume of accumulated history. If less than β of the total negotiation steps have elapsed, or if there has been no previous instance of successful negotiation, the required negotiation quantity is evenly distributed among all counterpart agents and that amount is offered. In this case, a minimum quantity of 1 is enforced. In all other cases—namely, during a negotiation step t where sufficient history has been accumulated—the distribution quantity is calculated first, followed by determining an adjustment value to establish the final allocated quantity.

The allocation quantity for agent i, $q_{t,i}$, is calculated by the following equation:

$$q_{t,i} = q_t^{\text{need}} \frac{\sum_{t'=0}^{t} \frac{t'}{T} \hat{q}_{i,t'}}{Z}. \tag{1}$$

Here q_t^{need} is the required negotiation quantity at step t, T is the total number of steps, $\hat{q}_{i,t'}$ is the quantity obtained from the successful negotiation in a past step t', Z is the distribution function, $\sum_i \sum_{t'=0}^{t} \frac{t'}{T} \hat{q}_{i,t'}$.

In this allocation scheme, agents with larger quantities achieved in past successful negotiations receive preferential allocations. Additionally, by assigning heavier weights to the most recent successful quantities in the history, the approach also accounts for responsiveness to changes in other agents' strategies.

The adjustment quantity $alpha_i(q_{t,i})$ for the allocation quantity $q_{t,i}$ is calculated using the following equation based on the historical offer acceptance rate:

$$\alpha_i(q) = \underset{\alpha' \in \{-1,0,1\}}{\operatorname{argmax}} \; P(i, q + \alpha'), \tag{2}$$

$$P(i,q) = \begin{cases} \frac{n(i,q)}{N(i,q)} & \text{if } N(i,q) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here $P(i,q)$ is the acceptance rate for quantity q by agent i, $N(i,q)$ is the number of times that quantity q was offered to agent i in the past, and $n(i,q)$ is the number of times that agent i accepted the offer of quantity q.

This adjustment factor is intended to modify the proposed quantity based on historical results, favoring quantities that have proven more likely to lead to successful negotiations. Finally, the offer is made using the best unit price, while the quantity is determined by the method described above.

2.2 The response strategy

AnalysisAgent responds to an offer from agent i based on the offered quantity q_i. Specifically, if the offered quantity q_i is below the value obtained by subtracting the expected successful quantity of other agents from the required negotiation quantity q^{need}, the offer is accepted; otherwise, it is rejected or the negotiation ends. The expected successful quantity is updated each time an offer is proposed to a negotiating agent. For example, when a quantity q_j is proposed to agent j, the probability that agent j will accept q_j (as defined in (Eq. 3)) is calculated. If this probability exceeds a certain threshold, q_j is added to the expected successful quantity; otherwise, q_j is subtracted from it. By using the expected successful quantity, the system can moderate its acceptance policy—it avoids being overly restrictive by optimistically assuming that all ongoing quantities will succeed, and it prevents over-acceptance that might result from a pessimistic assumption that everything will fail.

3 Evaluation

In the following experiments, we set n_configs = 30 and n_runs_per_world = 1. Table 1 presents the results of a preliminary experiment where negotiations were conducted using SimpleAgent, CautiousOneShotAgent (last year's champion), and SyncRandomOneShotAgent (the synchronous baseline agent), with their scores compared. The results confirmed that the SimpleAgent scores lower than agents based on the OneShotSyncAgent framework. Table 2 shows the results of a similar experiment in which SimpleAgent was replaced with AnalysisAgent. In this experiment, the parameter for AnalysisAgent is set to $\beta = 20$. Comparing Table 1 and Table 2, we can see that AnalysisAgent achieves good scores in the same environment as the preliminary experiment. AnalysisAgent consistently outperforms SyncRandomOneShotAgent in terms of score—a performance benchmark that SimpleAgent never managed to exceed—but it still falls short against CautiousOneshotAgent.

Table 1: The Result of Preliminary Experiment

Agent	Num. of days		
	50 days	125 days	200 days
SimpleAgent	0.985	1.024	1.003
SyncRandomOneShotAgent	1.040	1.049	1.029
CautiousOneShotAgent	1.087	1.098	1.079

Table 2: The Result of Main Experiment

Agent	Num. of days		
	50 days	125 days	200 days
AnalysisAgent	1.054	1.060	1.060
SyncRandomOneShotAgent	1.040	1.049	1.050
CautiousOneShotAgent	1.087	1.098	1.092

Additionally, Table 3 and Table 4 show, for each experiment, the frequency of steps in which discrepancies occurred between the quantity the agent is required to negotiate in one step and the quantity actually transacted in that step, expressed as a percentage. For example, Table 4 shows that 55% of all steps ended with a discrepancy of 0 relative to AnalysisAgent's required negotiation quantity.

Comparing SimpleAgent in Table 3 with AnalysisAgent in Table 4, it is clear that AnalysisAgent has significantly reduced the rate of steps where the discrepancy in required negotiation quantity is negative (<0). This indicates a decrease in the frequency of over-trading. Based on these results, it is concluded that AnalysisAgent has successfully alleviated the over-trading issue observed in SimpleAgent. On the other hand, compared to the aggressively accepting SimpleAgent, there is an increase in the rate of steps where the transaction

quantity falls short (by 1∼ 5 and 6∼10). This suggests that AnalysisAgent may be more conservative in its acceptance behavior.

Table 3: Percentage of Steps with Discrepency(Preliminary Experiment)

Agent \ Discrepancy	< 0	0	$1 \sim 5$	$6 \sim 10$
SimpleAgent	0.207	0.663	0.122	0.008
SyncRandomOneShotAgent	0.013	0.670	0.280	0.037
CautiousOneShotAgent	0.103	0.807	0.084	0.006

Table 4: Percentage of Steps with Discrepency(Main Experiment)

Agent \ Discrepancy	< 0	0	$1 \sim 5$	$6 \sim 10$
AnalysisAgent	0.029	0.550	0.407	0.014
SyncRandomOneShotAgent	0.031	0.636	0.278	0.055
CautiousOneShotAgent	0.128	0.749	0.110	0.013

4 Conclusions

We proposed the AnalysisAgent, which is an improved version of SimpleAgent, and demonstrated through experiments that in worlds where many agents adopt distribution strategies, the AnalysisAgent tends to achieve better performance than SimpleAgent.

However, while the proposed strategy of AnalysisAgent alleviates the over-negotiation issues observed in SimpleAgent, it introduces a new problem in the form of insufficient transaction quantity. To further enhance the utility of AnalysisAgent, it is necessary to refine its strategy so that it not only prevents over-trading but also fully executes transactions when the remaining required volume is low.