SolidAgent: An agent submitted to the ANAC 2021 SCM league

Soma Maeda[1], Takanobu Otsuka[2]
Nagoya Institute of Technology, Aichi, Japan
[1]maeda.soma@otsukalab.nitech.ac.jp
[2]otsuka.takanobu@otsukalab.nitech.ac.jp

August 10, 2021

Abstract

We created an agent named SolidAgent that limits the timing of negotiations to maximize profits. In my production strategy, the factory produces the products only when a contract is secured. My trading strategy uses a forecasting strategy, managing inputs and outputs using predictions in advance. My negotiation manager is a component that provides negotiation control functionality. SolidAgent only performs input negotiations in the first half. In the second half, my agent performs output negotiations. SolidAgent can secure materials in the first half and sell products in the second half.Therefore when negotiations are more active, it maximize profits.

1 Introduction

The SCM world simulates a supply chain consisting of multiple factories that buy and sell products from one another. The factories are represented by autonomous agents that act as factory managers. Each agent decides which other agents to buy and sell from, and then negotiates with them. Their goal is to turn a profit. The SCM world is very different from the real world. For example, the game is intended to further research on agent negotiation; as such the design emphasizes negotiation and de-emphasizes operations (e.g., scheduling). According to the game description, in the SCM world, profit is calculated as follows:

$$Profit = \frac{B_N + \frac{1}{2}I_N - B_0}{B_0}, \quad \cdots (1)$$

where B_0 and B_N are the factory's balances at the beginning and end of the simulation, and I_N is the value of the products in the factory's inventory at the end of the game. This value is based on the trading price, but to incentive trade, inventory is valued at only half the trading price. Therefore, we can

maximize profits by buying products cheaply and selling them at high prices, but we should not buy too many materials that we cannot sell because the price of surplus products is low. So we created SolidAgent by improving NegotiationManager, PredictionBasedTradingStrategy, DemandDrivenProductionStrategy. When SolidAgent is the buyer, it does not sign a contract with a very high price in order to maximize B_N. It also forecasts to ensure that it doesn't create surplus products.

2 Agent Strategy

SolidAgent limits the timing of the contract and signs a contract not to go bankrupt. Also it does not make high input contracts at a high price like a gamble, and make steady profits.

2.1 Agent Configuration

SolidAgent consists of the following four classes.

- SCML2020Agent
- MyNegotiationManager
- MyTradingStrategy
- MyProductionStrategy

"SCML2020Agent" is required class. "MyNegotiationManager" is based on "NegotiationManager". This negotiation manager is a component that provides negotiation control functionality to an agent. Based on this negotiation manager , SolidAgent limits the timing of the contract. "MyTradingStrategy" is based on "PredictionBasedTradingStrategy". This trading strategy uses a forecasting strategy, managing inputs and outputs using predictions in advance. "MyProductionStrategy" is based on "DemandDrivenProductionStrategy". This production strategy produces only when a contract is secured.

2.2 MyNegotiationManager

"MyNegotiationManager" is based on "NegotiationManager". It manages "MyNegotiator". "MyNegotiationManager" returns "MyNegotiator" when it offers or is offered negotiation requests and "MyNegotiator" manages the negotiation. "MyNegotiator" uses utility function to calculate the utility of the negotiation from the information like "price," "delivery date," and "quantity" related to the negotiation. It allows "MyNegotiator" to reject offer with late delivery time or more products than necessary, and try to make contracts higher prices when it is the seller and lower prices when it is the buyer. Also, "MyNegotiationManager" limits the timing of the contract. In the first half, it mainly performs input negotiations and collects materials. After that, it starts output negotiations.

It rejects output negotiations if it doesn't have enough input products in its inventry.

2.3 MyTradingStrategy

"MyTradingStrategy" is based on "PredictionBasedTradingStrategy". This trading strategy uses a forecasting strategy, managing inputs and outputs using predictions in advance. It predicts the price and quantity to buy and sell at each step. Given these two quantities, it maintains the amounts of inputs and outputs that it needs. It then employs a controller to manage negotiations and update the amounts secured. It also sets a threshold to avoid running out of money. Therefore, it prevents product shortages or bankruptcy due to over-contracting.

2.4 MyProductionStrategy

"MyProductionStrategy" is based on "DemandDrivenProductionStrategy". This production strategy produces only when a contract is secured. If we want to turn all materials into products, we should use "SupplyDrivenProductionStrategy". However, there are production costs when we produce products. So if we want to reduce the risk, we should use "DemandDrivenProductionStrategy".

3 Evaluation

We experimented with the run() function in the template to evaluate SolidAgent's performance. In addition to SolidAgent, DecentralizingAgent and BuyCheapSellExpensiveAgent are participating in this experiment, and we compare and evaluate their scores. The parameters are follows:

- competition = std,
- reveal_names = True,
- n_steps = 50,
- n_configs = 3,

and the scores of each agent are shown in the following table.

Table 1: Scores of each agent

Experiments	SolidAgent	DecentralizingAgent	BuyCheapSellExpensiveAgent
1	-0.06812	-0.125917	-0.18620
2	-0.0752474	-0.0741237	-0.30363
3	-0.0776338	-0.0836226	-0.42248
4	-0.0390445	-0.140561	-0.217341
5	-0.0738393	-0.092114	-0.368081
Average	-0.067977	-0.1032677	-0.299546

These results show that SolidAgent is the best of the three agents four out of five times. Comparing the average scores, we can see that SolidAgent is better than the others.

Conclusions

In this report, we explained SolidAgent's strategy. Also we showed that SolidAgent is better than some prepared agents by experiments. SolidAgent signs contracts based on timing and balance, but SolidAgent would score even higher if it could flexibly change prices for every situation. Also even if SolidAgent succeeds in signing a contract, it doesn't mean making profits. Because the signed contracts will be cancelled if other agent goes bankrupt. In the future, we would like to predict which agents are likely to go bankrupt, and improve the accuracy of predictions. We can create even better agents by using the prediction to optimize their behavior.