PrudentAgent: An agent submitted to the ANAC 2021 SCM league

Takeshi Wada[1], Takanobu Otsuka[2]
Nagoya Institute of Technology, Aichi, Japan
[1]wada.takeshi@otsukalab.nitech.ac.jp
[2]otsuka.takanobu@otsukalab.nitech.ac.jp

August 11, 2021

Abstract

The main concept of PrudentAgent is to reduce losses. To reduce losses can lead to higher scores in the SCML world. As for the contract, it depends on other agents, so we thought of a way to make a profit on my own. This report details the strategies implemented and their results.

1 Introduction

Supply chain management(SCM) is a management method that links the flow of goods and money with the flow of information to share and coordinate information throughout the supply chain for total optimization. Recently, SCM has been attracting attention due to corporate globalization and changes in the labor environment. So, we worked to create an agent that negotiate automatically and profitably.

In the SCML world, the reputation is determined by profit. The profit is calculated as follows;

$$profit = \frac{\sum_{a \in F} B_N(f) + \epsilon I_N(f) - B_0(f)}{\sum_{a \in F} B_0} \tag{1}$$

where, ϵ is the fraction of trading price at which to value the inventory at the end of the game. $B_0(f)$ and $B_N(f)$ are the factory's balances at the beginning and end of the simulation, respectively, and $I_N(f)$ is the value of the products in the factory's inventory at the end of the game.

As you can see from the equation, in the SCML world, monetary profit is evaluated regardless of the number of products sold or the number of contracts. PrudentAgent focuses on how to keep the balance positive.

2 The Design of PrudentAgent

From the above point of view, we made PrudentAgent which reduce waste in production processes and negotiations.

PrudentAgent consists of three strategies:

- MyProductionStrategy
- MyNegotiationStrategy
- MyTradingStrategy

In the next section, we will explain the strategies we inherited and the changes we made in each strategy.

2.1 MyProductionStrategy

We inherited DemandDrivenProductionStrategy to prevent overproduction of the products. This production strategy is designed to produce only what is needed for the contract that has been signed. In the SCML world, the products left in inventory are valued at half of the trading price at the end of the simulation. So, if we were to produce as many raw materials as we have, we might incur extra costs such as the production costs, which in turn might lead to losses.

2.2 MyNegotiationStrategy

The negotiation management strategy uses the MyNegotiationManager class that was introduced in the tutorial.

Here we have made one change. In order to prevent our products from remaining at the end of the simulation, we changed the program code so that the first half of the simulation is devoted to collecting raw materials and the second half to selling the products made. If the output is insufficient at the time of execution of the contract on the seller's side, it will result in a breach and we will suffer from large losses. This method is highly likely to prevent such an incident from happening in the first place.

However, the first half of the simulation is for collecting raw materials, so if you can't collect them by then, we won't be able to make products in the second half. This is one drawback.

2.3 MyTradingStrategy

For the trading strategy, we chose PredictionBasedTradingStrategy. This strategy uses an internal trading prediction strategy to predict the number of inputs that are expected to be available and the number of outputs that are expected to be sold by the agent at every time step.

Moreover, we modified the sign_all_contracts() function in this strategy slightly so that the agent cancels the contract when the execution of the contract would make our balance suffer significantly in the case of the buyer side. If a contract is such that our balance is less than 30% of it at the beginning of the simulation, the contract will be cancelled.

3 Evaluation

We ran 5 starndard track tournaments(nsteps=50, nconfigs=2) using run() method included in the template to evaluate PrudentAgent perfomance. We added MarketAwareIndependentNegotiationsAgent and DecentralizingAgent as competitors. The parameters are as follows:

- competition : std
- reveal_names : True
- n_steps : 50
- n_configs : 2

The results of the tournaments are shown in Table 1.

Table 1: Scores of 5 starndard track tournaments

Tournament	PrudentAgent	MarketAwareIndependent NegotiationsAgent	DecentralizingAgent
1	-0.0580621	-0.0750132	-0.143304
2	-0.0947145	-0.137575	-0.178678
3	-0.0663095	-0.11941	-0.243396
4	-0.0179488	-0.0299421	-0.170806
5	-0.0434309	-0.0498207	-0.174147
Average	-0.05609316	-0.0823522	-0.1820662

As you can see from the table, PrudentAgent has a higher score than the other agents. However, the scores themselves were all negative values. The reason for this is that the contract that PrudentAgent signed was not a good one for us, so we did not get much profit from it. From the above, PrudentAgent needs to make some modifications in the negotiation strategy.

4 Conclusions

As we mentioned at the beginning of this report, we focused on how to reduce losses of the agent and our approach may be somewhat profitable in the SCML world. However, in the real world, each factory will want to make a lot of profit. PrudentAgent described in this report only works in the SCML world, and it is unlikely to make much profit in the real world. It is necessary for us to think of negotiation and production strategies that can not only reduce losses but also increase profits a lot.

In addition, PrudentAgent contains several parameters (i.e. the range of amounts of money of the contract that PrudentAgent is requested), the values are set based on our own senses. We should have thought of more mathematical and logical approaches to set the values, but we couldn't do it this time. Such values may be also considered from a psychological point of view, so we would like to gain knowledge and think about it if we have time.