WTAgent: An agent submitted to the ANAC 2021 SCM league

Shuto Kawakita[1], Takanobu Otsuka[2]
[1,2]Nagoya Institute of Technology, Aichi, Japan
[1]kawakita.shuto@otsukalab.nitech.ac.jp
[2]otsuka.takanobu@nitech.ac.jp

August 11, 2021

Abstract

WTAgent maximizes profits by making the purchase price of inputs cheaper and the sale price of outputs more expensive. WTAgent's production strategy converts all input products to output products. WTAgent schedules as much production as possible to produce the required quantity before the time it is needed at. WTAgent's trading strategy uses prediction strategies to manage input products and output products that are needed. WTAgent's negotiation strategy is to buy a product for prices between 0 and half catalog prices, and sell a product for prices between catalog prices and double catalog prices.

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. The goal is to turn a profit, and the agent that turn the highest profit wins.

According to the game description in SCML2021, an agent's performance will be measured by its score. An agent's score will be the turncated mean of its profits in all simulations.
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. F is the set of all factories controlled by instantiations of the agent, $B_0(f)$ 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. So, the agent has to buy input products cheaper and sell output products more expensive.

2 The Design of WTAgent

WTAgent has three classes we created and one required class as follows:

- MyNegotiationManager
- MyTradingStrategy
- MyProductionStrategy
- SCML2020Agent

2.1 My Negotiation Manager

MyNegotiationManager consists of two steps.

First, We create our controller based on SAOSyncController. The main idea of our controller is that the controller will define a utility function for any possible outcome. It will collect offers from all controllers and responds in this way:

1. If the best offer is invalid, the controller rejects everything and offer the best offer.
2. If the best offer is within threshold, the controller accepts it.
3. Otherwise, the controller sends the best offer to all controllers else and tries to improve this offer until the end of the negotiation.

Our controller defines a utility function which is a linear combination of the price and difference between the quantity and WTAgent's needs at the delivery time.

Second, we use this controller in our negotiation strategy. Our negotiation control strategy will work as follows:

1. It will instantiate two SyncController objects. One is for selling. The other is for buying.
2. It will start negotiations to satisfy the needs that it gets from the trading strategy using these controllers every simulation step.

When negotiating, WTAgent buy a product between 0 and half catalog prices, and sell a product between catalog prices and double catalog prices.

2.2 OurTradingStrategy

MyTradingStrategy is based on PredictionBasedTradingStrategy. This strategy uses a trade prediction internally to predict how many inputs are expected to be available and how many outputs are expected to be sold by WTAgent at every time-step. Given these two quantities, it maintains the amounts of inputs/outputs that it needs. It then employs a controller to manage negotiations and update the amounts secured.

2.3 OurProductionStrategy

OurProductionStrategy is based on SupplyDrivenProductionStrategy that is converting all inputs to outputs. OurProductionStrategy is not only converting all inputs to outputs, but also prioritizing the production. For each signed contract, WTAgent schedules as much production as possible to produce the required quantity before the time it is needed at.

3 Evaluation

To evaluate the WTAgent's performance, we experimented with the run() functionpresent in the template. The parameters are as follows:

- competition=std
- nsteps=15
- nconfigs=3

The competitors of WTAgent were DecentralizingAgent, BuyCheapSellExpensiveAgent. The scores of each agent for the five experiments are shown in Table1.

Table 1: Score of 5 Starndard truck tournaments

Tournament	WTAgent	DecentralizingAgent	BuyCheapSellExpensiveAgent
1	-0.0641059	-0.238763	-0.939725
2	-0.160228	-0.632556	-0.908415
3	0.014639	-0.227615	-1.24358
4	0.0387795	-0.262767	-0.863824
5	-0.12575	-0.261178	-0.631781
Average	-0.05933308	-0.3245758	-0.9174642

Conclusions

In this report, we explained WTAgent's strategy. We thought buying inputs cheaper and selling more expensive were efficient way to turn profits. WTA-

gent's main strategy is buying inputs cheaper and selling outputs more expensive. At a glance, WTAgent is similar to BuyCheapSellExpensiveAgent, but WTAgent won BuyCheapSellExpensiveAgent. We think the cause of this is Negotiation Manager. Not only by buying inputs cheaper and selling more expensive, but also by scheduling as much production as possible to produce the required quantity before the time it is needed at, WTAgent could sign more contracts. Therefore, WTAgent won BuyCheapSellExpensiveAgent.