AgentSAS: An agent submitted to the ANAC 2022 SCM league

Shiraz Nave, Sariel Tutay, Amit Dayan
Bar Ilan University
shirazut@gmail.com, sariel362@gmail.com, 177amit@gmail.com

June 14, 2022

Abstract

We present a trading agent dedicated to the OneShot track. Our agent's strategy combines learning for predicting future market conditions and agents' behavior for optimal selection of concurrent offers. We improve our agent by monitoring the negotiating patterns of other agents, since these patterns can maximize our profits and minimize our expenses.

1 Introduction

The SCM OneShot world simulates a supply chain consisting of multiple factories that buy raw materials from, and sell final products to, one another. The factories are managed by autonomous agents. These agents are assigned a target quantity (drawn at random) to either buy or sell. They then negotiate with other agents to reach agreements, which become binding contracts that specify the terms of trade.

2 The Design of AgentSAS

2.1 Negotiation Choices

AgentSAS inherits from LearningAgent and therefore can get its price range of maximum limit for buying and minimum limit for selling. Instead of choosing those limits, we use an additional method, and attempt to learn the acceptance strategy of our partners in order to decide.

2.1.1 Saving history per negotiator

So far, we can choose our limits in the same way LearningAgent does. Different treatments are required for different patterns of negotiating. We try to keep track of the negotiators' behavior since this pattern can often maximize our profits and minimize our expenses. For this purpose, we keep for each agent their two latest offers for buying and selling.
Given agent i, its last two buying offers $lastBuying_{first}$, $lastBuying_{second}$ and its two last selling offers, $lastSelling_{first}$, $lastSelling_{second}$ we assume that the change in the next price will be similar to the change we saw before.

$$nextPrice_i^{buying} = lastBuying_i^{first} - (lastBuying_i^{second} - lastBuying_i^{first})$$

$$nextPrice_i^{selling} = lastSelling_i^{first} - (lastSelling_i^{second} - lastSelling_i^{first})$$

2.1.2 Choosing the maximum and minimum

Given an agent i, the price ranges chosen by the LearningAgent – $maxBuying_{learning}$, $minSelling_{learning}$, and the next assumed prices according to the agents' history, we determine our price limits as follows:

$$maxBuying_i = min(maxBuying_{learning}, nextPrice_i^{buying})$$

$$minSelling_i = max(minSelling_{learning}, nextPrice_i^{selling})$$

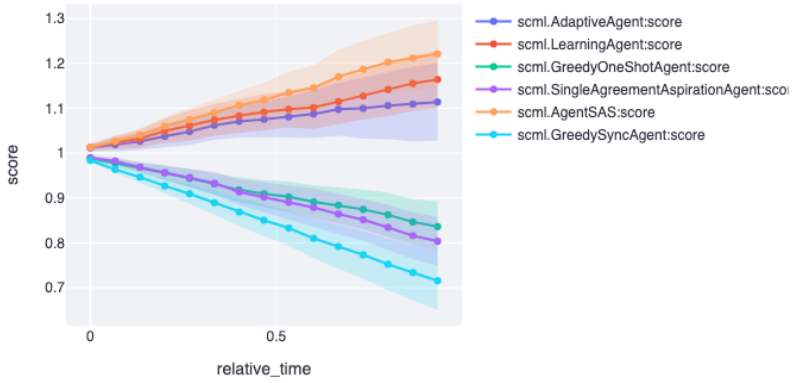
In that way we maximize our profit and minimize the cost according to agents' offering patterns.

2.2 Utility Function

We used the default built-in utility function provided by the SCML environment.

3 Evaluation

We can see that over time, the gap between the graph of AgentSAS and the other agents' graphs is increasing. Eventually, AgentSAS gets the highest score among all other agents.



Agents' scores over time via SCML visualizer

4 Conclusions

In this report, we described our agent, AgentSAS, which takes into account agents' behavior in order to determine its negotiation limits. AgentSAS also uses its ancestor methods in order to choose the best options for its own interests. Our agent performs better than the other agent and gets a higher score as shown in section 3. Improvements include optimizing the increase or decrease in trade prices in order to maximize our profits. Future work may include improvements in the evaluation of agents' behavior and their offering patterns.