CS 551 Introduction to Artificial Intelligence Project Report

Doğa Yılmaz - Ozan Tarlan

June 14, 2021

1 Introduction

Today we live in a world where every system we interact has some sort of automated component inside it. With the help of advancing technology the automation capabilities and levels of those systems are increasing every day. Within the scope of this project we are attempting to develop an automated factory management agent for the supply chain management league (SCML). SCML aims to simulate real-world factory management where multiple factories consist which can sell and buy products to each other. In the simulation factories are managed by autonomous agents. The main aim of the agents is to decide the actions of the factory in a way that the profit of the factory is maximized. The agent with the highest profit averaged over multiple simulations wins the Supply Chain Management League. [7]

In this report first we will give some background information regarding to the used technologies and techniques. Following that we will introduce our automated factory manager agent design which is named PolymorphicAgent. After that we will introduce the experiment setup and experiment results. Then, we will further elaborate on the agents which are contributed to the SCML 2020 and compare their results with our design.

2 Background

In this section we will provide some background information regarding to the technologies used in the scope of this project.

2.1 NegMAS

NEGotiation MultiAgent System (NegMAS) is a python library for developing autonomous negotiation agents as well as simulation environments. Aim of the library is to provide an environment for development of state of the art simultaneous and automated negotiation agents.[2]

2.2 SCML

The Supply Chain Management (SCM) world simulates a supply chain consisting of multiple factories that buy and sell products from one another.[7] SCML python library is developed for conducting the Supply Chain Management League. The SCML library is built on top of the NegMAS library and it uses the tools NegMAS provides. In this project our agent uses the provided tools from both of the libraries.

3 The Design of Polymorphic Agent

In this section we will elaborate on our automated factory manager agent which is developed for SCML. The agent consists of 3 main modules which are listed below. [3]

- A production strategy module which decides when to produce
- A trading strategy module which decides the quantities and prices of the products our agent sell or buy.
- A negotiation manager module which handles the negotiations between our suppliers, consumers and us.

Below, each module, corresponding sub-modules and the strategy behind them will be explained in detail.

3.1 Production Strategy

Our production strategy has 3 sub strategies. We divided the whole simulation timeline into 3 sections. In each section, our production strategy changes its behaviour. Below each sub strategy is elaborated.

3.1.1 Sub Production Strategy 1: Trade Driven

This sub training strategy schedules production only when there are some contracts which the agent did not initiate.

3.1.2 Sub Production Strategy 2: Demand Driven

This sub training strategy schedules production only when a contract is secured by the agent.

3.1.3 Sub Production Strategy 3: Supply Driven

This sub training strategy schedules production when there is some input product in the inventory. It keeps producing until the input product stock is exhausted.

3.1.4 Simulation Timeline Partitioning

As mentioned previously, we partitioned the simulation timeline into 3 partitions. The aim of this approach is to minimize the effect of drawbacks of each sub strategy by applying them when they perform the best. A sample partitioning can be found in the figure 1.

In our design there are 2 variables which controls the partition size (step count) of each partition.

3.1.5 Intended Production Behaviour

Our intended behaviour is to increase production in the early steps if there are some contracts our agent did not initiate. By doing so we aim that in the early steps our agent can more quickly produce when compared to demand driven methods which waits for a demand to be exist.

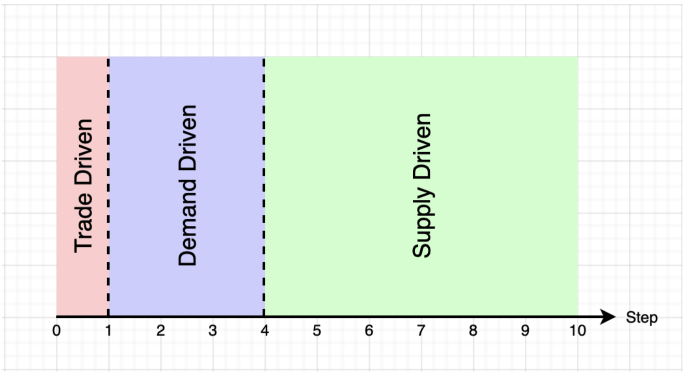

Figure 1: Partitioning of the simulation timeline

In the mid steps our strategy changes to a demand driven method due to expected decrease in the trade volume when compared to the early steps. The aim of this approach is not to overproduce throughout the mid steps.

In the steps which are close to the end of the simulation we switch our strategy to supply driven strategy. This approach decreases the possibility of finishing the game with some input product in the factory stock.

Algorithm 1: Our Production Strategy

```
On Step
if current step is greater or equals to partition threshold 2 then
    invoke supply driven step ;
end

On Contracts Finalized
if current step is smaller than partition threshold 1 then
    Apply trade driven sub production strategy ;
else if current step is smaller than partition threshold 2 then
    Apply demand driven sub production strategy ;
else
    Apply supply driven sub production strategy ;
end
```

3.2 Trading Strategy

In our design, the trading strategy adjusts the input cost of the required product we need to buy and the output price of the product produced by of our agent. The price adjustments are made according to market information provided by the SCML Agent World Interface. The price adjustments are made in a way such that the output and input prices matches the average price in the market.

Out trading strategy also keeps track on needed input products, secured input products as well as needed output products and secured output products. By doing so our agent can track the status of the missing products and determine the target input product quantity which will be bought later in the session. The pseudocode of our trading strategy can be found below.

Algorithm 2: Our Trading Strategy

```
On Step
Get average market prices for input and output products;
Update input cost and output price according to the retrieved market prices;

On Contracts Finalized
for contract in signed_contracts do
    if is_seller then
        update secured and needed outputs;
    else
        update secured and needed inputs;
    end
end
```

3.3 Negotiation Strategy

Negotiation is a crucial part of the SCML. By negotiating with our suppliers and consumers we propose contracts that include unit price, delivery time and product quantity that we want to buy or sell. In this section we will explain our negotiation approach which includes opponent modelling strategy, partner selection , utility function and acceptance strategy.

3.3.1 Opponent Modelling Strategy

In negotiation, having information about your opponent can be vastly beneficial for the success of the negotiation process. In the automated negotiation literature, collecting and analysing data about our opponent is called opponent modelling. In SCML 2021 we can obtain partial information about our opponents. The provided information about our opponents consist of their financial report. Financial reports in SCML include agent id, agent name, assets, breach level, breach probability, cash and bankruptcy information.

Our opponent modelling strategy is based on a "Trust" function which gives an overall value between 0 and 1 about the agents breach level and breach probability. Our "Trust" function can be found below.

$$T(n) = \begin{cases} 1 & \text{if opponent is BUYER or SELLER} \\ \frac{\sum_{0}^{current\_step} \frac{1}{1-(breach\_level * breach\_probability)}}{number\_of\_available\_reports} & \text{if opponent is other agents} \end{cases}$$

We update trust score for each of our suppliers and consumers each step. Our aim is to determine which of our opponents delivers the promised amount of products on promised time. Lastly we use the calculated trust points to select our partners, to adjust the utilities during negotiations and also in our acceptance strategy.

3.3.2 Partner Selection

In SCML selecting partners is also very important. If our supplier can not deliver the promised products on the determined time not only that agent receives a breach penalty but our production process might be disturbed due to lack of input items in our inventory. Thus, we prefer to negotiate and interact with the agents which have a high trust score in order to ensure that they will deliver the promised amount of products on time.

To ensure the reliability, our agent does not accept negotiation offers from agents which have a trust score below 0.15. An illustration of partner selection process can be found below.

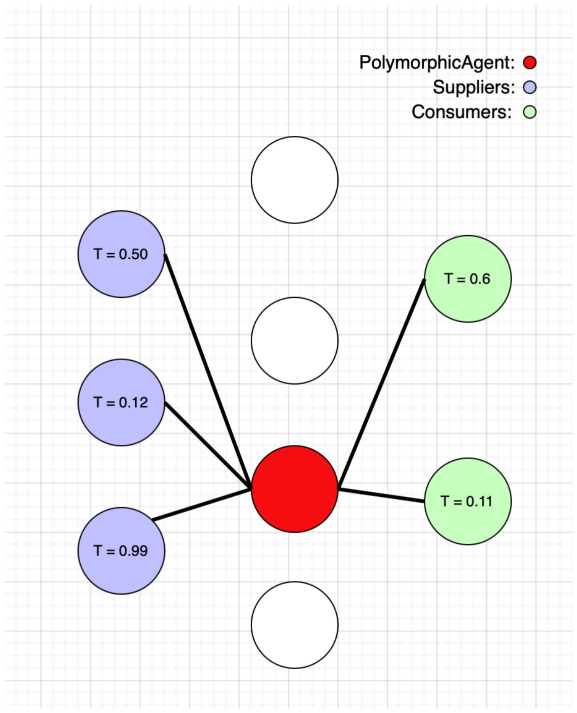

Figure 2: An illustration of partner selection process

As it can be seen from the figure above the supplier in the middle row and the consumer in the bottom row has a trust score below 0.15 thus, incoming negotiation requests from those agents will be directly rejected and any negotiation offer made by our agent will follow a stricter negotiation process.

3.3.3 Utility Function

As we have mentioned in the previous sections, our agent calculates trust score for every agent it might interact with. We used that information to adjust the utility function in a way that it will output a higher utility if the opposing party has a high trust score and a lower score for parties that have low trust score. Our modified utility function can be seen below.

$$U = \sum_{i=0}^{n_{outcomes}-1} w_i * \omega_i * OpponentTrust$$

3.3.4 Acceptance Strategy

Acceptance strategy is also an important part of our design. As discussed before, we update our input and output product prices according to the market information provided by the SCML library. We have set the minimum acceptable unit price of our input and output products to the the corresponding average market cost of that product. Also again as discussed before we have a minimum trust limit in our acceptance strategy.

4 Results

In this section the experiment setup will be explained in detail. Following that the tournament results of the tournament conducted between our agent and the SCML library agents will be presented.

4.1 Experiment Setup

Our experiment setup consists of a tournament where our agent and most of the agents in the SCML library competes. To run the tournament we used anac2021_std function provided by the SCML library. The parameters we used are listed below.

- n_steps = 100
- n_configs = 2
- competition = ”std”

Within the tournament, 60 sessions are conducted. The total tournament progress lasted 19 minutes and 4 seconds. The results of the tournament will be discussed in the experiment results section.

4.2 Experiment Results

In this section experiment results obtained using the experiment setup introduced in the previous section will be presented.

	Agent Type	**Score**
1	PolymorphicAgent	-0.00139626
2	DecentralizingAgent	-0.060915
3	MarketAwareDecentralizingAgent	-0.0829932
4	MarketAwareBuyCheapSellExpensiveAgent	-0.79044
5	BuyCheapSellExpensiveAgent	-0.79264

Table 1: Results of the conducted tournament between our agent and the agents in the SCML library

As it can be seen from the table above, PolymorphicAgent outperforms all of the agents in the SCML library. Considering the naive approach of most of the library agents, the results were as expected. More comprehensive experimentation of our agent with the top performing SCML 2020 participants will be introduced after a brief discussion of their strategies in the following section.

5 Related Work

In this section top performing 6 SCML 2020 agents will be summarized. Following that tournament results which contains our agent as well as all of the listed agents below will be presented.

5.1 SCML 2020 Agents

5.1.1 Merchant

Merchant is one of the participant agents in SCML 2020. The Merchant agent tries to explore and exploit other agents by sending abnormal offers. If one of the agents accepts the abnormal offer further negotiation requests are sent which the merchant agents acts like a tough negotiator. Furthermore, the agent does not accept any negotiation offer in order to prevent exploration.[9]

5.1.2 MMM

MMM Agent is yet another agent which participated to the SCML 2020. The MMM agent mainly focuses on Risk management and Opponent modelling. MMM sends a negotiation request for all of its suppliers and consumers. If the request is responded positively, MMM recognizes the sender as an ally. By doing so MMM aims to model its opponents and tries to make offers to them which might be beneficial for both parties.[8]

5.1.3 SteadyMgr

The SteadyMgr Agent initially purchases the maximum number of materials available for production and produces as much output product as possible. And by actively trading the produced items it aims to make a lot of profit.[5]

5.1.4 SavingAgent

The SavingAgent aims to minimize the production costs of the output product. The agent searches for the optimal production schedule by considering the contracts that have been concluded, the inventory quantity of input and the cost of the production line.[6]

5.1.5 BARGentCovid19

The BARGentCovid19 agent is based on the combination of base agent strategies and a smart sorting mechanism which distinguishes between the sellers which wants higher price and the buyers which want lower price. After the sorting operation, the agent tries to avoid starting negotiation with those parties.[1]

5.1.6 ASMASH

The ASMASH agent is based on a decentralization agent. The motivation of the agent is to reduce the legitimacy of the partner after declaring bankruptcy.[4]

5.2 Tournament Results With SCML 2020 Agents

We conducted a tournament among our agent and the top performing 6 agents from SCML 2020. The conducted tournament consists of 560 sessions and each session is 100 steps long. The total run time of the tournament was 1 hour and 45 minutes. As it can also be seen from the table below our agent outperforms SteadyMgr, SavingAgent, BARGentCovid19 and ASMASH.

	Agent Type	**Score**
1	Merchant	0.0165519
2	MMM	-0.000251835
3	PolymorphicAgent	-0.00517554
4	SteadyMgr	-0.0196771
5	SavingAgent	-0.0524061
6	BARGentCovid19	-0.0667796
7	ASMASH	-0.0753374

Table 2: Tournament results of the tournament conducted between our agent and 6 top performers form SCML 2020

6 Conclusion

As a conclusion first we have suited the problem domain end environment that our agent will use. After that, we have designed and implemented an automated factory management agent for SCML. Following that we have tested our agent with the built in agents of the SCML library. Finally, we have summarized the top performing agents from 2020 and also conducted a tournament which contains our agents and the 2020 top performers. According to our experiment results our agent outperforms the SCML library agents. Moreover, according to the further experiments we have conducted, our agent has a comparable performance to the top performing agents in the SCML 2020.

References

[1] Agent bargentcovid19: An agent submitted to the anac 2020 scm league.

[2] NegMAS documentation. http://www.yasserm.com/negmas/. Accessed: 2021-06-13.

[3] SCML documentation. http://www.yasserm.com/scml/scml2020docs/. Accessed: 2021-05-21.

[4] Assaf Marsha et al. Asmash: An agent submitted to the anac 2020 scm league.

[5] Masahito Okuno et al. Steadymgr: An agent submitted to the anac 2020 scm league.

[6] Takuma Kawamura et al. Savingagent: An agent submitted to the anac 2020 scm league.

[7] Y. Mohammed et al. Supply chain management league automated negotiating agents competition.

[8] Kazuki Komori. Agent mmm for scml, anac 2020.

[9] Ayan Sengupta. Agent merchant for scml, anac 2020.