Designing and Implementing a Negotiating Factory Manager
Group 1 Project Report

Cem Ata Baykara
Department of Computer Science
Ozyegin University
Istanbul, Turkey
ata.baykara@ozu.edu.tr

Alp Demirezen
Department of Computer Science
Ozyegin University
Yalova, Turkey
alp.demirezen@ozu.edu.tr

***Abstract*—This report includes the implementation details and strategies on the project of CS 551 Introduction to Artificial Intelligence course. The aim of the project is to design and implement a negotiating factory manager agent on the Supply Chain Management League (SCML) environment [1] which is an automated negotiating agents competition. In this project we aim to implement novel approaches which includes acceptance, production, trade prediction strategies. We discuss how these ideas, when implemented, can affect the performance of the agent and provide their implementation details.**

I. Introduction

In this report, we will be designing our own agent to compete in Supply Chain Management League (SCML). SCML is an environment that provides a healthy world for negotiation agents to compete in. To compete in this environment the agents in the SCML world need to perform a variety of complex tasks and take important decisions autonomously. These actions and decision of the agents are governed by their respective strategies which will be discussed further in Section III. The novel contributions and ideas proposed in this paper are as follows:

1) A market aware production strategy which improves on the existing supply and demand driven production strategies. The proposed production strategy can alternate between supply and demand based production strategies based on the trading price of its input and output products as well as its production cost.
2) A machine learning based trade prediction strategy in which the agent predicts it expected input and output products for each step using a pre-trained regressor.
3) A novel signing strategy which evaluates the value of each contract to be signed at a given time step. And after ordering these contracts based on their calculated value, starts signing contracts starting with the highest valued contract until the agent satisfies its current needs.

The rest of this report is structured as follows. Section II will provide the background information about the SCML world and competition environment. Section III will present the anatomy of a SCML agent, mainly how the agent performs autonomous tasks required to compete in the environment. Section IV will discuss and present the novel ideas proposed in this paper and Section V will talk about their implementation details. Section VI will present the experimental setup we used to test the performance of our agent with the proposed improvements and present the results and performance of our agent. And finally Section VII will include our conclusions and possible future scope of our agent.

II. Novel Ideas

The novel ideas proposed in this report are threefold, which include the following strategies:

- Production Strategy
- Trade Prediction Strategy
- Signing Strategy

This section will describe the ideas proposed in detail.

A. Market Aware Production Strategy

There are three different production strategies available in the SCML environment by default. These are the supply, demand and trade based production strategies as presented in section III. We believe that Using only one of these production strategies at all times is not an optimal production strategy which optimizes the performance of the agent.

The supply based production strategy may make the agent lose value if the price of its output product is lower than the sum of the price of its input product and its production cost. In such cases the agent will still produce if it is using a supply based production strategy making the agent lose value. Of course this strategy may be preferred depending on the other strategies employed by the agent, however we consider production strategy individually in this section.

Similarly, the demand based production strategy is also not an optimal production strategy which can be adapted at all time steps. In this case, if the output trading price of the agent is actually higher than the sum of its input product price and production cost, the agent can generate value by producing. However if the agent is employing a demand based production strategy, the agent will only produce based on its

signed contracts and therefore may miss the opportunity to generate this value by just producing.

We believe that a more optimal production strategy is a hybrid production strategy, alternating between supply and demand based production strategy based on the market prices. The proposed production strategy normally employs a demand based production strategy if the price of the agents output product is lower than the sum of its input product price and production cost. In such a case producing may make the agent lose value and therefore the agent only produces if it needs to sell based on its ongoing contracts. If however the price of the agents output product is higher than the sum of its input product price and production cost the agent switches to a supply based production approach since it can generate value by just producing its inputs. Since this strategy includes the available market information to switch between the production strategies in the form of input trading prices and output catalog prices for a specific agent, we named this strategy as the market aware production strategy.

B. Modified Signing Strategy

We propose a novel approach on the signing strategy. Currently, the default sign_all_contracts function orders all the contracts for the agent based on the contract times before considering which contracts to sign. This essentially allows the agent to consider the contracts which are closer to the present time step. Even though this is a nice approach, since as the contract time increases farther into the future, it makes it difficult for the agent to forecast its expected income and needs and plan beforehand effectively. However we believe that just considering time for the ordering of the contracts before deciding on which ones to sign is not the most optimal approach. There may be a highly valuable and promising contract in the future and since the contracts are ordered by time in default, the agent may already have satisfied its needs with sub optimal contracts and miss the opportunity to sign high value contracts.

We propose an evaluation function for the contracts which evaluates the expected value based on the available information of the contracts. And instead of ordering the contracts based on their times for signing, our proposed method orders them based on their values for the agent. By doing this we essentially force the agent to consider high valued contracts first to satisfy its needs. And after the agent satisfies its current needs from the available contracts the agent stops signing.

The challenging part of this approach was to came up with an accurate formula to calculate the value of each contract. We implemented this function differently for buy and sell contracts as follows where P_U denotes the unit price of the contract, T_C and T_S denotes the contract time and current time step respectively, and P_O and P_I denotes the output trading price and input trading prices respectively.

$$V(C \mid s) = \frac{P_U}{(T_C - T_S + 1)P_O}$$

The formula above is used to calculate the value of the contract given that the agent is the seller. The time of the contract with respect to the current time has a negative correlation with the value of a contract as one might expect. And the fraction of P_U and P_{OT} gives an estimation of how much extra profit we are getting by selling from this unit price with respect to the current trading price of our output product.

$$V(C \mid b) = \frac{P_I}{(T_C - T_S + 1)P_U}$$

The formula above is used for calculating the value of a contract where the agent is the buyer. The formula is quite similar other than the fact that the unit price on the contract and its respective trading price has changed places. Now the agent values how much cheaper it can actually buy from the contracts. The time of the contract still has a negative effect on the value of its value.

C. Machine Learning Based Trade Prediction Strategy

We propose a trade prediction strategy based on machine learning for estimating the amount of input and output products to expect in the future based on available information to the agent. We have collected data from many worlds which include a random pool of agents to help sample data from agents which behave differently. Our agent collected its own data during the training phase using the final version of all other modification which we have implemented to help make the collected data as accurate and as unique to our agents behaviour as possible. Using the pre-trained regressor, our agent performs trade predictions in real-time.

III. Implementation Details

A. The Anatomy of our Agent

As mentioned in Section III, agents need to employ some components to perform better in the SCML environment. In this section, We will be providing information about the components and strategies that our agent employs. Either the ones that we implemented ourselves, or the ones that we used from default components.

Our agent utilizes a machine learning based trade prediction strategy, a market aware production strategy, a machine learning based trading strategy, an execution rate prediction strategy and a step negotiation manager. We have modified all of these components, however some of the changes are minor ones which makes the default strategies we have used compatible with our modified strategies. Therefore we would like to introduce them to you before we start presenting the implementations of the novel ideas.

Execution rate prediction strategy(ERP), is a component that specifies a ratio for a contract to be signed between two agents. There are two types of ERP strategies that are provided by the SCM environment. We have implemented minor changes on Mean_ERP strategy. The default ERP strategy uses a static execution rate of 0.5 for all opponents. We have modified this to be able to calculate our expected quantities from different opponents better. Our modified ERP strategy keeps separate

execution rates for each of our opponents dynamically, and updates these rates based on the behaviour of its opponents.

A negotiation manager is an essential component of any agent since deciding how to negotiate and under what conditions is important for agents to be successful. There are a variety of negotiation managers provided by default in the SCM environment. Step negotiation manager is one of them. Two controllers, one for selling and another one for buying, are created at each step, and these controllers manage the negotiations for each time step. While doing so, the controllers use an average execution fraction rate. Since we have individualized execution rates for each of our opponents we implemented an algorithm that takes the mean of these execution rates, rather than having a static average execution fraction rate for every opponent.

B. Market Aware Production Strategy

As mentioned in Section IV, we have implemented a dynamic production strategy, which means agent will adopt a supply driven production strategy if it is worth to produce. And a demand driven strategy if producing is not logical. Our agent decides on how to act based on the trading price of the input product and the catalog price of the output product. The pseudo code can be seen below.

Algorithm 1: Market Aware Production Strategy

P_C : Production cost
if $output\_price/2 > input\_price/2 + P_C$ **then**
  Adopt a supply driven strategy.
else
  Adopt a demand driven strategy.

C. Machine Learning Based Trade Prediction Strategy

We have trained a random forest regressor for our proposed machine learning based trade prediction strategy. We use the default set of hyper-parameters while training our regressor and did not performed hyper-parameter tuning to achieve the best results and therefore there is a lot of room for improvement. Using the pre-trained regressor, the agent tries to predict its expected input and output products for the given horizon. In our implementations we took the horizon as $current\_step+1$, so the agent just tries to predict its expected products for the immediate next step. We have used the random forest regressor available in scikit-learn [3] library. For the input features of our regressor we have used the following information available to the agent.

1) my_input
2) my_output
3) relative_time
4) n_competitors
5) n_processes
6) n_products
7) current_balance
8) current_input_inventory
9) current_output_inventory

We generated our own data to train our regressor model. The data we have generated includes many SCML world runs where a our agent collected its own data to make the collected data as unique and accurate for our agent as possible.

D. Modified Signing Strategy

The implementation of the modified signing strategy proposed in Section IV in details was rather simple. The only function which we have changed is the sign_all_contracts function in the trading strategy. The pseudo code of the modified signing strategy can be seen below.

Algorithm 2: Modified Sign All Contracts

Input : C_N : Contracts to be signed
Output: S_N : Signed contracts
$P_I = input\_trading\_price$
$P_O = output\_trading\_price$
$V_N = dict()$
$s = current_step$
for C_i *in* C_N **do**
  $is\_seller =$ annotation of C_i
  $u =$ unit price of C_i
  $t =$ time of C_i
  if is_seller **then**
    $value = u/((t - s + 1)P_O)$
  else
    $value = P_I/((t - s + 1)u)$
  end
  $V_N[C_i] = value$
end
sort V_N based on values in descending order
sign until needs are satisfied using sorted V_N return S_N

After ordering the current contracts to be signed based on their values, the proposed signing strategy proceeds like the default sign_all_contracts function. The agent starts considering the contracts starting from the highest valued contract currently available and keeps signing until it satisfies its current input and output needs.

IV. Results

This section will present our results and experimental setup which we have used to measure the performance of our agent implemented with the proposed ideas.

A. Experimental Setup

To measure the performance of our agent we have generated an environment which includes two of the most powerful agents available by default in the SCML environment namely; the market aware decentralizing agent and the market aware independent decentralizing agent. The world also include the winner agent of the ANAC 2020 SCM League, the SteadyMgr agent [2] which is a very powerful agent. Using these three powerful agents and our agent we have made a simulation

run which includes 30 SCM worlds to measure the average performance of our agent.

B. Performance of the Agent

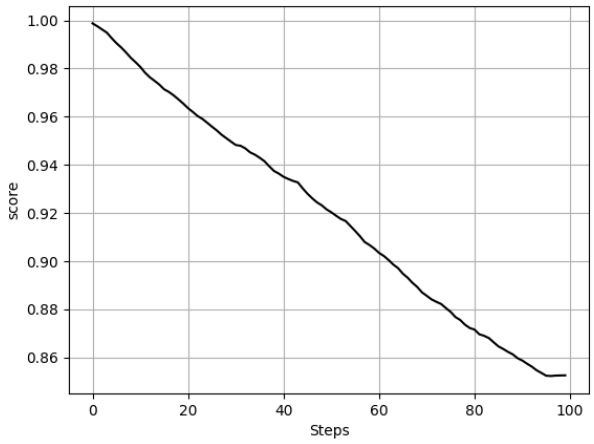

Fig. 1. The average score of our agent throughout each simulation step.

The average score of our agent throughout each simulation step can be observed from Fig. 1. As expected the score of the agent steadily decreases at each step, however this is an expected behaviour of a SCM agent. Thus, a score of about 0.85 against strong agents is quite promising.

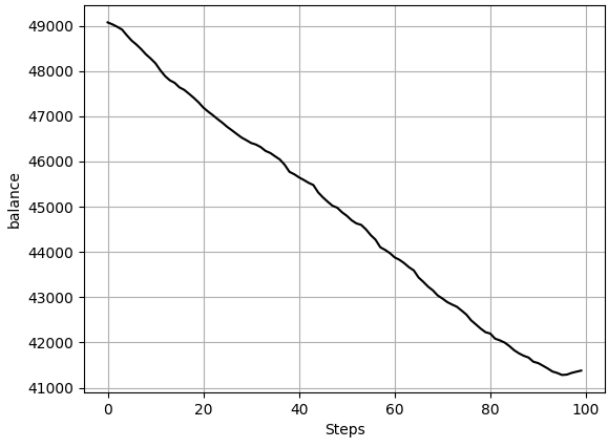

Fig. 2. The average balance of our agent throughout each simulation step.

The average balance of our agent in the experiment we performed can be seen from Fig 2. Since the score of an agent has an extremely high correlation with the agents score, the Fig.'s 1 and 2 are quite similar.

The average assets of our agent throughout each simulation step can be observed from Fig. 4. It is clear to see that our agent has a problem with selling the output products it produces as its assets appear to increase steadily at each step. Currently we believe that this may be due to the fact that we made only minor modifications for the negotiation strategy and we are using the default step negotiations manager. Another reason for this behaviour is that the proposed agent may be

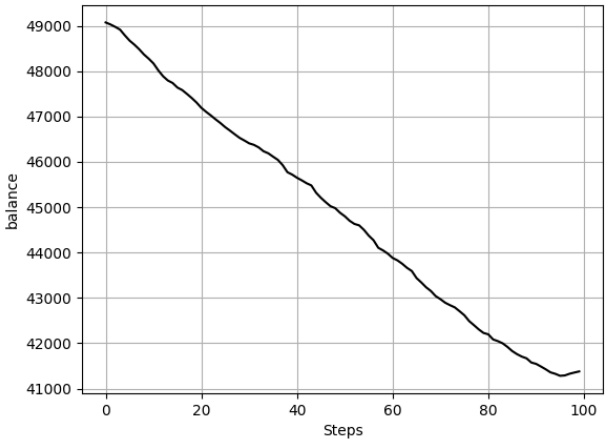

Fig. 3. The average balance of our agent throughout each simulation step.

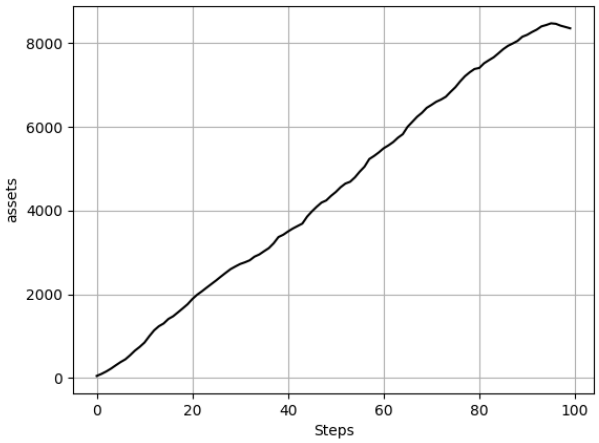

Fig. 4. The average assets of our agent throughout each simulation step.

unable to come to an agreement when negotiating to sell its output product, because due to our modified signing strategy we know that the agent should sign contracts even if it needs to sign sub optimal contracts to be able to always satisfy its needs. Nonetheless, this is also an aspect which needs improving for our proposed agent.

You can observe the average productivity of our agent throughout each simulation step from Fig. 5. From Fig 5. it can be concluded that our agent produces at around 0.65 of its production capacity during the middle of the simulations and produces less at both the beginning and at the end of the simulation.

The average spot market quantity bought by our agent throughout each simulation step can be seen from Fig. 6. The spot market usage of our agent appears to be approximately 0.48 based on our experiments, which is a good indication that our agent is not using the spot market very much. We believe that this may also be a good indication that our machine learning based trade prediction strategy appears to be working well. Similarly you can observe the average spot market loss of our agent throughout each simulation step in our experiments

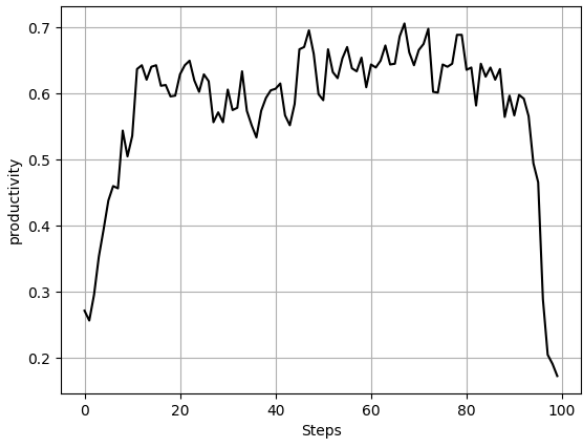

Fig. 5. The average productivity of our agent throughout each simulation step.

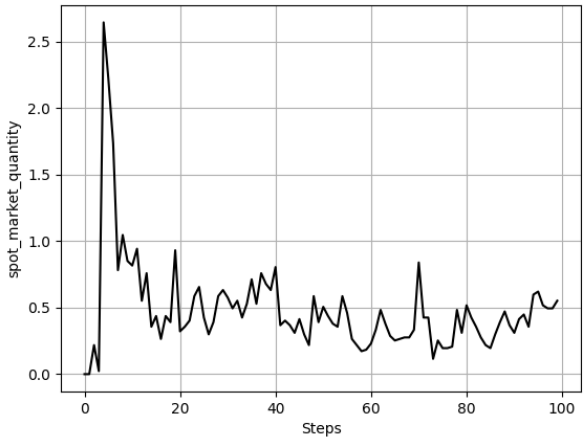

Fig. 6. The average spot market quantity bought by our agent throughout each simulation step.

from Fig. 7.

TABLE I
AVERAGE PERFORMANCE OF OUR AGENT AT THE END OF EACH GAME MEASURED OVER 30 WORLD RUNS

Performance Metric	Value
Average Score	0.8526
Average Balance	41379.58
Average Assets	8358.63
Average Productivity	0.1724
Average Spot Market Quantity	0.4855
Average Spot Market Loss	0.9562

The average performance metrics of our agent measured against the powerful agents stated above and using the experimental setup presented above can be seen from Table 1. Overall we are quite satisfied with how our agent performed against strong opponents like step based and independent market aware decentralizing agents, as well as the winner of the last years ANAC SCM league. We believe that getting a score of 0.85 is not a bad performance against such strong opponents.

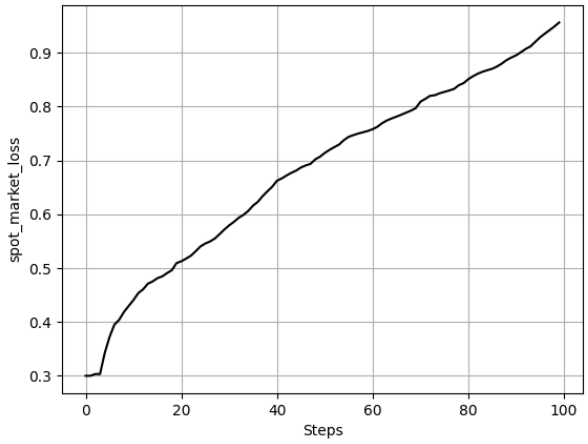

Fig. 7. The average spot market quantity bought by our agent throughout each simulation step.

You can observe the performance of our random forest regressor trained on our data generated over many world runs from Table 2. Currently even though our R2 score is very high, the MSE of our model is also quite high, this may be due to the fact that the regressor over fitted to our data.

TABLE II
PERFORMANCE METRIC OF OUR RANDOM FOREST REGRESSOR USED FOR THE PROPOSED MACHINE LEARNING BASED TRADE PREDICTION STRATEGY

Performance Metric	Value
R2 Score	0.9846
Mean Absolute Error	7.0932
Mean Squared Error	393.39

There are a variety of improvement which can be made to our agent. Firstly, the machine learning based trade prediction strategy can be improved to include a comprehensive research to decide which regressor or model is best fit for this current problem case. We decided to use the random forest regressor without conducting such a deep experimental research. The training can also be improved as we did not performed any hyper-parameter tuning to increase the performance of our model.

The proposed signing strategy can also be improved by modifying the contract evaluation function which we have proposed to include more data about the contract to make it more accurate. This formula can also be improved and adapted to agents differently, depending on their behaviour and what the agent values more.

V. CONCLUSIONS

In this project, we implemented an agent capable of performing in the SCM League environment. We presented three different novel strategies, their implementation details, as well as how they can improve the performance of an agent in the SCM environment. We have employed some of the default components that are provided in environment, as well as some novel ideas we came up with to construct an agent that is

unique. We incrementally implemented our ideas observing the performance of our agent throughout each step to see how the modifications will affect its accuracy. We have also presented an experimental setup which we have used to measure the overall performance of our agents final version against powerful agents available in the SCM environment by default, as well as the winner agent from ANAC SCM League 2020. The results show that our agent performs well even against strong opponents but there are still a vast amount of strategies and components that can be improved.

There are more features that must be implemented for our agent to negotiate with a human-being. First of all we have to utilize well-known negotiation strategies. Our negotiation manager should be implemented thoroughly calculating best negotiations properly for every step. We also need to train our model with a bigger dataset. Fine tuning of our parameters is also essential to maximize our agent's performance. Last but not least, our agent requires an interface to be able to interact with people if we would like to perform human vs. machine negotiations.

REFERENCES

[1] Y. Mohammed, A. Greenwald, K. Fujita, M. Klein, S. Morinaga, and S. Nakadai. Supply chain management league, automated negotiating agents competition, 2020.

[2] Masahito Okuno and Takanobu Otsuka. Steadymgr: An agent submitted to the anac 2020 scm league, 2020.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.