YIYAgent: An agent submitted to the ANAC 2021 SCM league

Yehoshua Stern[1] Yoel Benabou[1] Itay Cohen[1]

[1]Bar Ilan University, Israel

1 Introduction

The main idea behind YIY agent is determining the needed quantities and prices for each time step based on past observations. This idea is considered on both trading and negotiation strategies. In addition, we employed a few heuristics which suppose to yield better deals with other agents.

2 The Design of YIYAgent

The YIYAgent class is based on of the following three classes:

- YIYTradingStrategy
- YIYNegotiationManager
- YIYProductionStrategy

Each class represents a strategy component.

2.1 Trading Strategy

The trading strategy is responsible for deciding the quantity and price for buy and sell at every time-step. It specifies some goal which later the Negotiation Manager will try to fulfill. In our approach, we decided to try to improve upon PredictionBasedTradingStrategy by using a trained scikit-learn model to predict the expected input and output, using the released data. We Implemented a class SklearnTradePredictionStrategy, inheriting from TradePredictionStrategy, to perform the predictions. TradePredictionStrategy allows for predicting an initial prediction for each timestep in the beginning, and then allows for refining the prediction at every step. Therefore, we used two models, one that predict the values (expected input and output) from just product number and timestep, to use in the initial prediction, and one that predict the values based on short history of recent values, for refining the estimates at each timestep.

2.1.1 Data and Models

We used the data released by the competition organizers, which was gathered from 5000 tournaments. Specifically, we used the stats.csv files which contain detailed statistics for every day in the simulation. For the initial prediction we trained a model for each product to predict values from day of the tournament as a fraction between 0 and 1. For the refining prediction we trained based on the day and previous values. For both kinds of models, we tried several model types and chose a random forest, because it is appropriate for data such as this, which does not have any obvious metric on it, and because it is easier to find good hyperparameters for. To show the results, here are some plots of actual and predicted values for "sold quantity", for some tournaments:

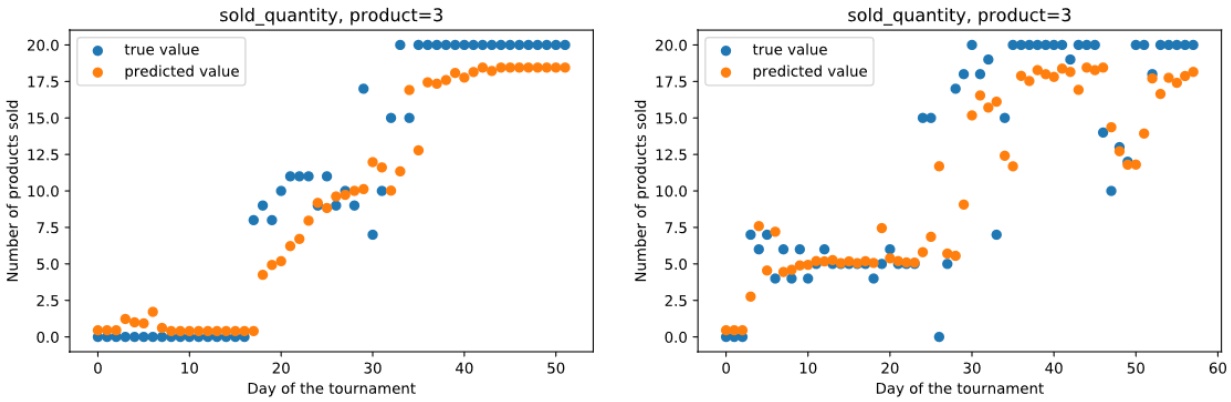

Figure 1: actual and predicted values for "sold quantity"

We can see the models can predict the values at around the true values and are approximately correct for most of the days.

2.2 Negotiation Strategy

Our agent uses a negotiation strategy which extends an existing strategy - MovingRangeNegotiationManager. This existing strategy uses an SAOSyncController to handle multiple negotiation offers synchronously. The main idea of this strategy is to offer negotiations with a relatively naive price ranges which are set with respect to the catalog prices. Then, this strategy handles multiple negotiations synchronously, selects the best offer and tries to improve it, while all the other agents recieve a counter offer which is equivalent to the best offer received before. An agreement is made if an offer passes a fixed relative threshold, or if the negotiation is close to an end. We changed this strategy in a few ways. First, we set a static breach level threshold. We will not start a negotiation with an agent if its breach level is above the threshold. The idea behind this heuristic is to increase the probability of a non violated agreements. Second, we set a fixed horizon of two time steps in the step() function of our negotiation manager. Finally, we kept an history of selling and buying prices based on successful negotiations (agreements). This price history is managed by a sligthly modified SAOSyncController. When it is time to set selling and buying price ranges for future negotiations, those ranges are based on an average of the last three selling and buying prices respectively. We as-

sume that basing our future negotiation on successful past negotiations may result in better future agreement (with respect to the utility function of MovingRangeNegotiationManager)

2.3 Production Strategy

Our agent uses an hybrid production strategy. It will apply a supply driven strategy if the half of the production output price is greater than the half of the input price + the cost production price. Else, it will apply a driven strategy to prevent producing unneeded products, thus saving in production cost.

3 Evaluation

YIYAgent was tested against Decentralizing Agent and BuyCheapSellExpensiveAgent several times for 50 time steps. In most of the runs YIYAgent achieved the highest scores among the three agents.

4 Suggestions

- Some of the static thresholds such as breach level threshold, time threshold, utility threshold could have been set to be dynamic, as the variance of different simulations may be relatively high.
- A heuristic for late sells should be considered. For example, a mechanism which gradually decreases the selling price as the simulation is getting close to an end.

5 Code

The code for this project can be found below [1].

References

[1] (2021) Yiy-agent - supply chain management league. [Online]. Available: https://github.com/itay99988/scml_project