# SkyAgent: An agent submitted to the ANAC 2022 SCM league

Natsumi Sakamoto[1], Takanobu Otsuka[2]
Nagoya Institute of Technology, Aichi, Japan
[1]sakamoto.natsumi@otsukalab.nitech.ac.jp, [2]otsuka.takanobu@nitech.ac.jp

June 16, 2022

**Abstract**

We created an agent named SkyAgent. SkyAgent is an Agent that decides whether to sign a contract considering the average purchase cost and average production cost. Also, SkyAgent relaxes the requirements of contract towards the end of the game, like real-world inventory disposal. Therefore, SkyAgent is an agent that doesn't sign a disadvantageous contract and reduces the final factory inventory as much as possible.

## 1 Introduction

In the real world, factories determine the price of their products by taking into account the costs in producing, buying, selling, and managing manufactured products. Based on that minimum price, prices are determined so that profits are maximized. However, it is difficult to make profits excessively large. Because of the existence of competing companies and budgets for those with whom we negotiate to conclude contracts.

There is a significant difference between the real world and the SCML world. In the real world, labor and management costs are incurred every day, so it is necessary to make a profit by signing a contract, even if the contract is quite unprofitable. In contrast, in the SCML world, if you don't sign a contract, you don't have to pay. In other words, there is no need to sign a disadvantageous contract.

Based on this difference, our agent, SkyAgent, takes into account the average cost of the bought products and the average production cost, and signs only profitable contracts. Also, the remaining inventory in the factory at the end of the game results in a lower score. Because of that, we have incorporated a real-world inventory discount by relaxing the requirements for signing the contract toward the end of the game.

## 2 The Design of SkyAgent

SkyAgent consists of three strategies:

- SkyProductionStrategy
- SkyTradingStrategy
- SkyNegotiationsManager

It also inherits TradingStrategy, ProductionStrategy, NegotiationManager and SCML2020Agent.

SkyProductionStrategy is based on SupplyProductionStrategy. SkyStrategy is a production strategy that converts all inputs to outputs.

## 2.1 SkyTradingStrategy

SkyTradingStrategy is based on PredictionBasedTradingStrategy.

SkyAgent decides whether or not to sign a contract according to the following risk management rules.

### Seller risk Management

For risk management, two conditions are defined for deciding whether or not to sign a contract.

a. **Dairy minimum sallling price update**

The first condition is that the minimum selling price is changed depending on whether or not there were sales on the previous day, and that no contract is signed below the minimum selling price. The minimum selling price on the first day is the price in the catalog. Starting the next day, if there are sales on the previous day, the minimum selling price is raised by 5% (max:twice the catalog). In contrast, if there are no sales on the previous day, the minimum selling price is lowered by 5% (min:price of the first day).

In this way, the minimum selling price fluctuates depending on sales, and only more profitable contracts can be signed. Also, it prevents the lack of sales for a long period of time.

| | $quentity\_sold_{i-1} > 0$ | $quentity\_sold_{i-1} = 0$ |
|---|---|---|
| $step = 1$ | $price_i = catalog\_price$ | |
| $step > 0$ | $price_i =$<br>$\min(price_{i-1} \times 1.05, catalog\_price \times 2)$ | $price_i =$<br>$\max(price_{i-1} \times 0.95, catalog\_price)$ |

$quentity\_sold_{i-1}$ : quentity of output on the previous day
$catalog\_price$ : catalog price of output
$price_{i-1}$ : minimum selling price of output on the previous day

b. **Average of input_cost and production_cost**

The second condition is whether the price of the contract is lower than the sum of the average buying cost untill the previoius day and the production cost by AgentInformation. The average buying cost untill the previous day is as follows :

$$average\_buying\_cost_i = \frac{\sum_{k=1}^{i-1} (input\_cost_k \times quentity\_bought_k)}{\sum_{k=1}^{i-1} quentity\_bought_k} \tag{1}$$

$average\_buying\_cost_i$ : average of input prices (step=i)
$input\_cost_k$ : price of input (step=k)
$quentity\_bought_k$ : quentity of input (step=k)

By not signing into contracts that are lower than the sum of the average buying cost and the average production cost, it is possible to sign into only those contracts that generate a net profit.

Also, if the factory has inventory at the end of the game, the score will be lower. Therefore, in order to be more profitable in the first half of the game, we sign only contracts that yield a net profit of at least 10% are signed.

```
if $step < 0.5$ :
    $contact\_price < (average\_buying\_cost_i + production\_cost) \times 1.1$
                                                        $\Rightarrow$ No contract
else :
    $contact\_price < (average\_buying\_cost_i + production\_cost)$
                                                        $\Rightarrow$ No contract
```

### Buyer risk Management

For risk management, two conditions are defined for deciding whether or not to sign a contract.

a. **Dairy maximum buying price update**
The first condition is that, as with Seller, the maximum buying price is changed depending on whether or not there were sales on the previous day, and that no contract is signed above the maximum buying price. This effect is the same as 1.2.1, so it is omitted.

| | $quentity\_bought_{i-1} > 0$ | $quentity\_bought_{i-1} = 0$ |
|---|---|---|
| $step = 1$ | $price_i = catalog\_price$ | |
| $step > 0$ | $price_i =$ $\max(price_{i-1} \times 0.95, catalog\_price)$ | $price_i =$ $\min(price_{i-1} \times 1.05, catalog\_price \times 2)$ |

$quentity\_bought_{i-1}$ : quentity of input on the previous day
$catalog\_price$ : catalog price of input
$price_{i-1}$ : maximum buying price of input on the previous day

b. **Prohibition of end-of-game buying**
The second condition is that the game must not be near its end. because even if it is bought, it will not be sold in the game and will remain in inventory. For this reason, we don't buy after 80% of the game.
if $(step > 0.8) \Rightarrow$ No contract

## 2.2 SkyNegotiationsManager

### Negotiation Choices

SkyNegotiationsManager is based on IndependentNegotiationsManager. IndependentNegotiationsManager is a negotiation manager, according to the tutorial, that manages independent negotiators that do not share any information once created.

We give SkyNegotiationManager an acceptable unit price for buying and selling products from other Agents. The acceptable unit price changes every day depending on whether or not there was a buy or sell on the previous day.

When Seller, the acceptable unit peice of output on the first day is 110% of the catalog price. Starting the next day, if there was sales on the previous day, increase the acceptable unit price of output by 10%, otherwise return to the price on the first day.

When Buyier, the acceptable unit peice of input on the first day is 90% of the catalog price. Starting the next day, if there was buying on the previous day, decrease the acceptable unit price of input by 10%, otherwise return to the price on the first day. Also, if the factory has inventory remained, the score will be low. Therefore, when it is after 60% of the game period, the acceptable unit price is the price on the first day. And, when it is after 90% of the game period, the acceptable unit price is not set.

By setting the rate of price fluctuations in negotiations to 10% and the rate of price fluctuations in risk management to 5%, it is possible that boldy negotiate while taking care of the risks.

### Utility function

The utility function of SkyNegotiationManager is set to LinearUtilityFunction. LinearUtilityFunction is a function provided by NegMas.

if Seller:

$LinearUtilityFunction(1, 1, 10)$

if Buyer:

$LinearUtilityFunction(1, -1, -10)$

## 3 Experiments

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

- competition=std
- reveal_names=True
- n_steps=50
- n_configs=2

We added DecentralizingAgent, MarketAwareReactiveAgent and IndependentNegotiationsAgent as competitors The scores of each agent for the five experiments and their means are shown in Table 1. This table shows that SkyAgent has the best score all five times.

| Experiments | SkyAgent(MyAgent) | DecentralizingAgent | MarketAwareReactiveAgent | IndependentNegotiationsAgent |
|---|---|---|---|---|
| 1 | **-0.04903** | -0.19524 | -0.23715 | -1.84054 |
| 2 | **0.01054** | -0.12845 | -0.22014 | -1.29236 |
| 3 | **0.03075** | -0.09038 | -0.57126 | -1.22007 |
| 4 | **0.02137** | -0.32992 | -0.29616 | -1.69489 |
| 5 | **-0.00251** | -0.36731 | -0.13941 | -1.38401 |
| Average | **0.00313** | -0.27870 | -0.29282 | -1.48637 |

Table 1: Scores of experimental results

## 4 Conclusions

This report describes SkyAgent.

SkyAgent has two main features. The first feature is to sign a contract by taking into account the average buying cost and the average production cost. The second feature is relaxation of requirements such as real-world inventory disposal. As a result, SkyAgent showed a better score than other agents in Chapter 3.

However, SkyAgent scores are not always positive. Because, SkyAgent decides whether to sign contact only from the price of contact. Therefore, it is necessary to have a negotiation strategy that decides whether to sign a contract based on the contract deadline and quentity of contracts.