ArtisanKangaroo: An agent submitted to the ANAC 2021 SCM league

Koki Katagiri[1], Takanobu Otsuka[2]
Nagoya Institute of Technology, Aichi, Japan
[1]katagiri.koki@otsukalab.nitech.ac.jp
[2]otsuka.takanobu@nitech.ac.jp

August 4, 2021

Abstract

The Main concept of ArtisanKangaroo is not using utility function in negotiations. Even if one selling offer has a very high utility value, it will end up with a breach unless the agent has enough input products and production capacity before its delivery time. ArtisanKangaroo also keeps track of every signed contract to prevent the agent from commiting breaches. With these features, we tried to minimize loss and outperform agents using utility function.

1 Introduction

In the real supply chain, suppliers and consumers negotiate with each other about how many products they trade, how much they cost, and when the contract is going to be executed. While they are negotiating in person, they both often compromise to some extent based on patners' response because companies and factories have periodic fixed cost such as personnel expenses and maintenance cost, and they need to make contracts continuously so that they won't be in the red.

However, in the SCML World, there are some differences from the real. First of all, one agent doesn't have to pay at all if it doesn't sign any agreements. This means that there is no incetive for agents to compromise and sign contracts regularly. Secondly, negotiations may end suddenly and the agent can't know the reason. If the unitprice of an offer is too expensive, its partner may terminate the negotiation rather than giving a counter offer. Finally, using linear utility function in negotiations can result in the agent's loss. Since linear utility function adds up a quantity, unitprice, and delivery time with multiplying weights, there are some possible combinations of those three factors which have the same utility value. However, agents' production capacity, balance, and products in the inventory are limited during a simulation. Therefore, the offer whose properties are far from the agent's asset should be less valued. This can be the problem if one's agent depends on utility function.

2 The Design of ArtisanKangaroo

From perspectives described in introduction, We made the agent ArtisanKangaroo which doesn't use utility function in negotiations or compromise to make contracts.
ArtisanKangaroo consists of two main components:

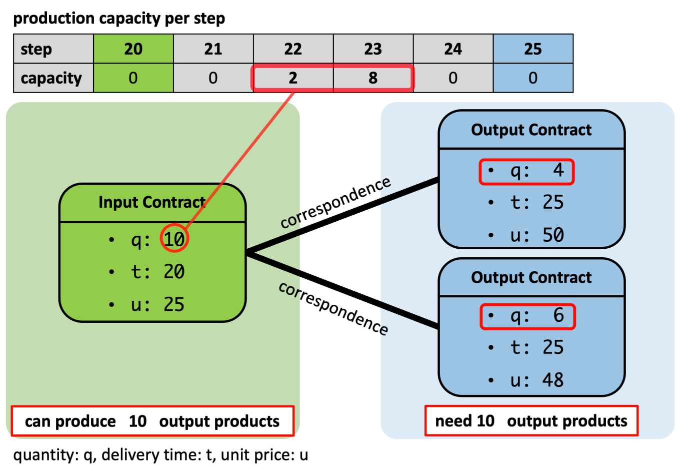
- MyTradingStrategy
- MyNegotiator

It also inherits SupplyDrivenProductionStrategy and SCML2020Agent. We chose SupplyDrivenProductionStrategy for the production strategy because signing output contracts after making output products hardly causes breaches.

2.1 MyTradingStrategy

MyTradingStrategy is responsible for managing contract correspondence: which input contracts will be used for an output contract and which contracts don't have corresponding contracts yet. During this process, it also considers whether it is possible to produce enough output products from the input contracts until output contracts' deadline. Thanks to this tracking, it is possible to detect how many input products are available and how many output products are required in a specific step.

Figure 1 shows a example of correspondence. The quantity of the input contract is equal to the total quantity of two output contracts, and there is enough production capacity to produce 10 output products before the delivery time of the output contracts. Therefore, it will make correspondence between the input contract and two output contracts.


production capacity per step

step	20	21	22	23	24	25
capacity	0	0	2	8	0	0

Figure 1: Making correspondence between input and output contracts

2.2 MyNegotiator

When ArtisanKangaroo requests negotiations or accepts other agents' negotiation requests, it returns MyNegotiator which is a marionette of ArtisanKangaroo. In the response phase, MyNegotiator simply checks if properties of received offer fulfill all of the predefined profitable conditions and accepts it. The conditions are intended to filter out disadvantageous offers like too large a quantity or late delivery time. In the propose phase, MyNegotiator proposes a offer whose property values were all passed from ArtisanKangaroo. Regarding those values, the quantity is fixed to n_lines because a contract to trade too many products at one time is more likely to cause a breach. The delivery time will be dynamically changed according to the partner agent's offer, and the unitprice depends on the catalog price and the number of previous successful contracts. Though the value of the unitprice which MyNegotiator proposes changes through a simulation, it does not change during a negotiation. Therefore, MyNegotiator proposes offers with the same unitprice in one negotiation, which means MyNegotiator doesn't compromise. This is how ArtisanKangaroo negotiates with other agents without utility function.

2.3 Risk Management

To minimize loss ArtisanKangaroo may receive, we dealt with three major risks;

- a surplus of input products
- a shortage of output products
- other agent's bankrupt

Two Former risks are related to negotiation and sign tasks. In the negotiation phase, MyNegotiator accepts offers which satisfy all of constraints to make profit, but it doesn't consider the execution possibility because there is no penalty for not signing agreements in the sign phase. At this point, one of the most important part of ArtisanKangaroo is sign_all_contracts(). Basically, ArtisanKangaroo signs input contracts whose delivery time is within first 40% steps, and signs output contracts only if it has already made input contracts which can be applied to them. In other words, it tries to buy input products first, then sell output products produced from them, and keep track of correspondence between input and output contracts. This trick prevents the agent from signing too much output contracts and commiting breaches. In addition, we introduced the maximum numbers of products in stock, which is helpful to avoid excess stock and leads to reduce loss.

Successfully making contracts don't always mean making profit. This is because the signed contracts can be cancelled if the partner agents go bankrupt. Since ArtisanKangaroo knows contracts correspondence, it can find which contracts will be affected in those cases. Accordingly, it tries to buy or sell products to compensate the cancellation.

2.4 Collusion Strategy

Our collusion strategy is eliminating negotiations between two ArtisanKangaroo. If ArtisanKangaroo receives negotiation requests from other ArtisanKangaroo, MyNegotiator will be switched to collusion mode. In collusion mode, one MyNegotiator proposes the offer whose unitprice is the same as the catalog price, and the other MyNegotiator accepts it with no check. At the same time, ArtisanKangaroo also negotiates with other agents, and choose contracts to sign out of all agreed contracts. This process allows ArtisanKangaroo to have guaranteed contracts and making profit stably.

3 Evaluation

We ran 5 starndard track tournaments(n_steps=50, n_configs=5) using run() method included in the template to evaluate ArtisanKangaroo's perfomance. We added DecentralizingAgent and MarketAwareIndDecentralizingAgent as competitors. The results of the tournaments are shown in Table 1. It shows that all of ArtisanKangaroo's scores are positive and around 0.05 and all of two competitors' scores are negative. ArtisanKangaroo's Average score is much higher than two competitors' as well. From these results, it can be said that ArtisanKangaroo which doesn't use utility function performed better than agents using utility function.

Table 1: Scores of 5 starndard track tournaments

Tournament	ArtisanKangaroo	MarketAwareInd DecentralizingAgent	DecentralizingAgent
1	0.0572293	-0.316561	-0.648157
2	0.0525736	-0.0602493	-0.355871
3	0.0345108	-0.122012	-0.342691
4	0.0457349	-0.218503	-0.385981
5	0.0680915	-0.0690787	-0.240927
Average	0.05162802	-0.1572808	-0.3947254

Conclusions

We considered differences between the real supply chain and SCML World, then described ArtisanKangaroo. Our evaluation result suggests that agents do not necessarily use utility function to make profit and the strategy that the agent doesn't compromise is effective in SCML. However, we limited the agent's flexibility such as negotiation quantity and delivery time, so there is a possibility that the agent misses opportunities to buy or sell more products. If these parameters are changed adaptively in the simulation, more profit is expected.