SCML 2020

Tamar Stern and Haim Nafcha

Strategy

In our project, we decided to take the prebuilt strategies and improve multiple parts of them.

We run multiple testing and chose the leading strategies.

Trading Strategy

For our Trading Strategy we chose the prebuilt *PredictionBasedTradingStrategy.*

We decided to change this trading strategy and we implemented the following trading limitation:

Blocking breaching agents

On incoming negotiation requests - if the agent of this contract has breached in the last n steps, we will block the contract.

In our experiments, when we tried to tune the steps number to look for breaches, we didn't find significant improvement between n=20 and n=unlimited, but on the live tournament we saw some improvement in the standard tournament.

We also checked dynamic breach blocking - we will be more forgiving to breaches in proportional to the game steps. The reason is that as the game progresses, it makes more sense that some agents will breach contract from time to time after some steps.
We ran this agent alongside the same agent with static breaching agents blocking (of all agents that breach in the last 20 steps) and with some additional built-in agents (DecentralizingAgent, BuyCheapSellExpensiveAgent, RandomAgent, DoNothingAgent, IndDecentralizingAgent and IndependentNegotiationsAgent), and we saw better results for the agent with static breaching agents blocking, so we didn't implement this method.

We implemented the breaching limitation on incoming contracts only, because when we tried the same limitation on start negotiations, the results compared to the same agent with incoming contracts filtering only were worse.

We thought of several reasons for this result:

1. If we limit both sides, our agent don't have enough contracts to produce, so we tried to limit the breaching agents for less steps back, so we will have more contracts, but this didn't improve the results either.
2. The improvement we saw when we blocked the incoming contracts from breaching agents were not because we blocked the breaching agents, but because we limited the number of incoming contracts, so we tried to limit the number of all incoming contracts only without checking if the agent have breach or not, but again, the results of the breaching agents block were better.
3. We tried to block breaching agents on outbound contracts only but not on incoming contracts, but this gave us worse results than blocking breaching agents on incoming contracts only.

Preferring contracts from other instances of our agent

For every incoming contract, we checked if we have other instance of our agent that offer the same product, and if we found one, we didn't sign on this contract in order to save the slot for our agent.

The problem with this method is that it is possible that we will cancel incoming contract to save slot for another instance of our agent, but this instance will sign contract with another agent.

If we will apply this to outgoing contracts it should solve some of this problem, but we still must manage the amounts of contracts, because it's possible that only one of our agents have this product to offer, but because of him we will block the contracts to many of our other agents.

As with blocking breaching agents, preferring contracts from other instances of our agent give best results on incoming contracts only, that strengthen the hypothesis that limiting the incoming contracts can help in some way, probably the combination of limiting incoming contracts but not all the incoming contracts, with filtering some bad contracts can help the agent.

Production Strategy

We the prebuilt *SupplyDrivenProductionStrategy,* and we didn't improve the agent in this area.

Negotiation Manager

For our Negotiation Manager we used *StepNegotiationManager.*

This kind of negotiation manager creates controllers named *StepController.*

The regular controller uses linear utility function called *LinearUtilityFunction*, we chose to change this to *NonlinearHyperRectangleUtilityFunction* with exponential function, and got better results.

We used it if we are sellers and if we are buyers too.