Agent30: An Agent Submitted to the ANAC 2020 SCM League

Authors: Guy Heller:hellerguyh@gmail.com, Elisha Gerson egerson96@gmail.com, Idan Hen Idan.hen@gmail.com, Matan Akrabi matanel@gmail.com - Bar Ilan University

Introduction - The main idea behind Agent30 is to calibrate the agent demand for input and output product such that at the end of the match the agent will have no inventory left on the one hand, and on the other hand it will buy and sell as much as possible with a reasonable profit. Agent30 followed the general structure provided by the SCML library. The agent is separated into 3 strategies that work together: Trading, Negotiation and Production. The brain of the agent is the Trading Strategy which dictates how much it needs to buy and how much it needs to sell in each time step.

Trading Strategy - Risk Management

In order to reduce the risk for selling or buying too much the agent updates it `inputs_needed[t]` and `outputs_needed[t]` in run time. An agent needs for output and input products depend on the game environment (e.g. amount of players). Before the game start the agent assumes some a-priori distribution ('`baseline function`') of buy and sell needs and it updates them as the game advances. Agent30 indication for mismatch between its a-priori distribution and the real needs is expressed as expected inventory at each time step of the game. Following is Agent30 algorithm for calculating the expected inventory and output and inputs needs . `SI[t]` - secured* inputs at time t ; `SO[t]` - secured outputs at time t; `IN[t]` - inputs needed at time t `ON[t]` - outputs needed at time t; `EI[t]` - expected inventory at time t; `NLINES` - number of production lines; `s` - current step; `n` - number of steps in simulation; `baseline` is described Baseline Function section

$$EI[t] = \sum_{i=0:s} f(SI[i] - SO[i]) + min(0, \sum_{i=s+1:n} SI[i] - SO[i])$$
$$f(x) = max(x, 0)$$

$$\alpha[t] = (EI[t]/2 * NLINES)$$
$$\beta[t] = max(\exp(1-\alpha), exp(0))$$
$$IN[t] = \beta[t] * baseline_{in}[t]$$

$$\gamma[t] = \max(0.1, EI[t]/NLINES)$$
$$ON[t] = \gamma[t] * baseline_{out}[t]$$

*Inputs and outputs are considered secured if a contract for buying (selling) these products was already signed.

Baseline Function - For baseline outputs Agent30 uses a constant value which is the amount the factory can make in a single time step. For baseline inputs, Agent30 baseline function prefers to buy more sooner then later.

$$baseline\_inputs[t] = 3.5 * NLINES * (1 - \tanh(\frac{1.6t}{n}))$$

Production Strategy

Agent30 uses an hybrid production strategy. It starts with a supply driven strategy which ensures it will have enough output products and changes after 80% of the game to demand driven strategy that prevents producing unneeded products, thus saving in production cost.

Negotiation Choices

Simultaneous Sell Negotiations Coordination - Agent30 implements the `SyncController` strategy for coordination between sell negotiations with the following differences - Expected Inventory Based Acceptable Output Unit Price, Modified Utility and Allowing Good Offers.

Expected Inventory Based Acceptable Output Unit Price - Agent30 acceptable output unit price is the minimum price Agent30 uses to start a sell negotiation. This value depends on the expected inventory in the time of negotiation as detailed here. `C` - production cost; `EI[t]`, `NLINES` - described in previous sections; `IC[t]` - input cost at time t; `AOC[t]` - acceptable output unit price at time t; `IC[t]` - price of input product at time t

$$\alpha[t] = \frac{EI[t]}{NLINES}$$
$$\beta[t] = \begin{cases} 1.2 & \text{if } \alpha[t] < 0 \\ 1 & \text{if } \alpha[t] \leq 4 \\ 1 - \alpha[t] * \frac{1}{80} & \text{if } \alpha[t] \leq 8 \\ 0.9 & \text{otherwise} \end{cases}$$
$$AOC[t] = (C + IC[t])\beta[t]$$

Modified Utility - This utility will make sure that an offer will get positive utility only if it gives profit of at least 20% over the cost of input product and production `AVG_IP` - average input price, `offer.q` - quantity in offer; `offer.p` - offer price; `offer.t` - offer time

$$f(q) = min(0, \frac{ON[t] - (q + SO[t])}{ON[t] - (q + SO[t])})$$
$$P(p) = p - (AVG\_IP + C) * 1.2$$
$$Utility(offer) = f(offer.q) \times [P(offer.p) \times 0.95 + 0.05 \times offer.q]$$

Allow Good Offers - A case in which non of the offers suggested to Agent30 have positive utility is a hint for the controller that its expectation is too high, and it should respond with a low utility offer. To do so the controller will check the negotiation boundaries for each of the negotiations and find for each the offer with smallest price that still achieves positive utility. These offers will be suggested to the negotiations partners.

Buy Negotiation Strategy - Agent30 buy negotiation strategy is based on `StepNegotiationManager` strategy with two differences: Expected Inventory Based Acceptable Input Unit Price and Modified Aspiration Calculation on Respond

Expected Inventory Based Acceptable Input Unit Price - Agent30 acceptable input unit price is the maximum price in which Agent30 accepts a buy negotiation. This value depends on the expected inventory in the time of negotiation as detailed here. `C` - production cost; `EI[t]`, `NLINES` - described in previous sections; `OP[t]` - spot output price at time t; `AIC[t]` - acceptable input unit price at time t; `AVGOP` - average price of output products

$$\alpha[t] = \frac{EI[t]}{NLINES}$$
$$\beta[t] = \begin{cases} \frac{(6 - \alpha[t]/2)}{5} & \text{if } \alpha[t] < 2 \ \wedge \ \alpha[t] < 10 \\ 0.1 & \text{if } \alpha[t] \geq 10 \\ 1 & \text{otherwise} \end{cases}$$

$$AIC[t] = \begin{cases} (OP[0] - C) \times \beta[t] & \text{if t<5} \\ (AVGOP * 0.9 - C) \times \beta[t] & \text{otherwise} \end{cases}$$

Modified Aspiration Calculation on Respond - Agent30 negotiator is based on the `AspriationNegotiator`. The `AspirationNegotiator` aspiration ranges from the minimum utility for the agent for the current negotiation range, `ufun_min`, to the maximum, `ufun_min`. Agent30 changes the upper limit to the following

$$\text{ufun\_ok} = utility(q = 1, t =, 1p = OP[0] - C)$$
$$\text{ufun\_max'} = \text{ufun\_max} \times 0.3 + \text{ufun\_ok} \times 0.7$$

Evaluation

Agent30 was tested against Decentralizing Agent in multiple environments which included 2-3 process and 1-3 agents per process. In most of the trails Agent30 achieved higher score then the Decentralizing Agent, and in all of the trails it received positive score.

Suggestions

The agent uses many hyper-parameters in it's models (used to calculate outputs/inputs needed, acceptable prices etc). A more sophisticated agent would have used Q-Learning to decide upon these parameters.