Automating Concurrent Negotiations in Supply Chain Management

g-RIPS Sendai – NEC Group
Team Members: Isaac Brown, Soto Hisakawa, Soichiro Sato, Kevin Zhou
Academic Mentor: Masaki Ogawa, Ph.D.
Industry Mentor: Yasser Mohammad, Ph.D.

Outline

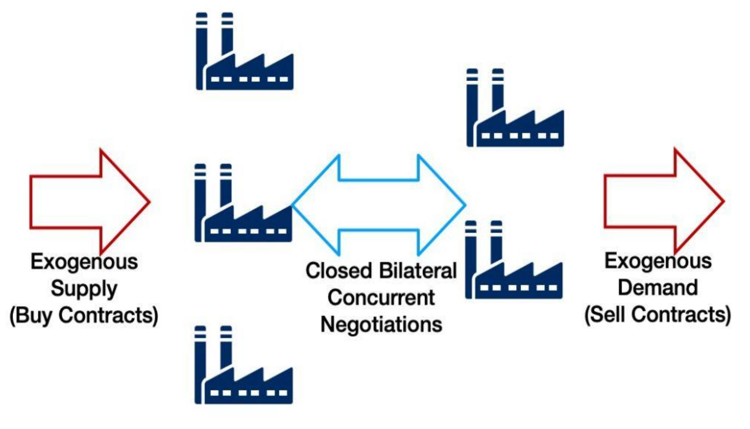
1. **Introduction**
    1.1. **SCML Game**
    1.2. **Problem/Goal**

2. **Reinforcement Learning**
    2.1. **Background**
    2.2. **PPO**
        2.2.1. **PPO agent**
    2.3. **Q-Learning**
        2.3.1. **Q-learning agent**

3. **Conclusion**

OneShot SCM Framework

- Agents each control a factory in the supply chain.



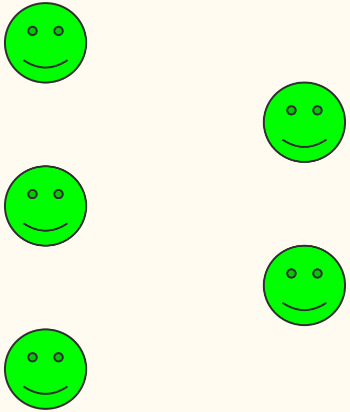
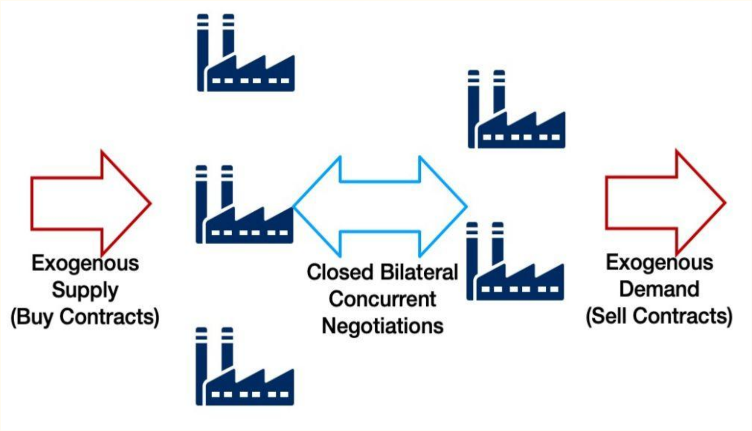


Image source: http://www.yasserm.com/scml/scml2020docs/tutorials/01.run_scml2020.html

OneShot SCM Framework

- Agents each control a factory in the supply chain.
- Each day in the SCM world, the following occurs:
  - Exogenous contracts are distributed



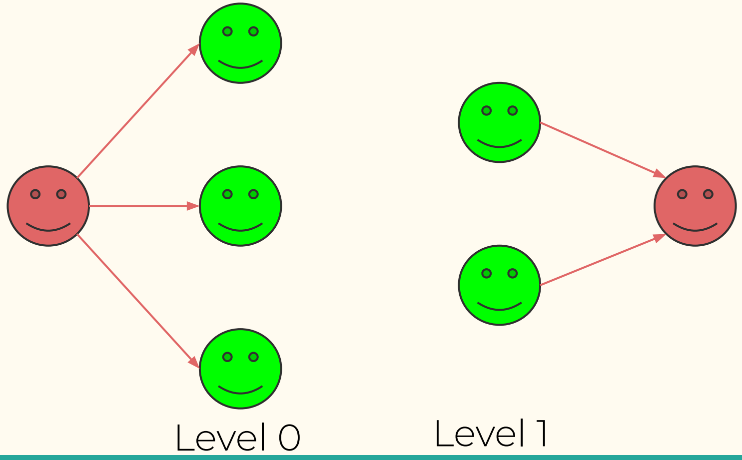
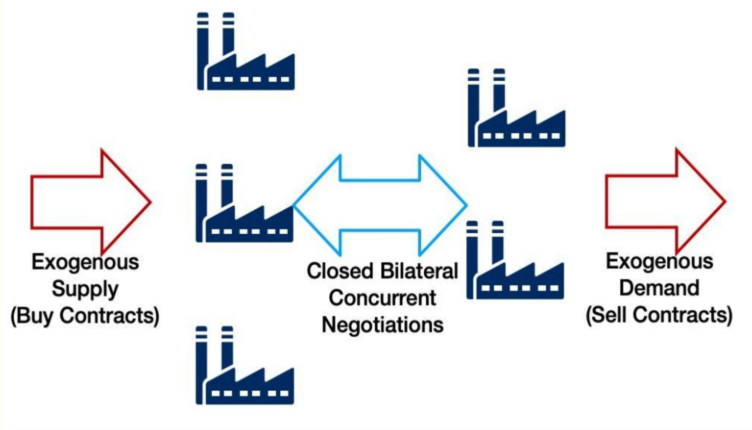


Image source: http://www.yasserm.com/scml/scml2020docs/tutorials/01.run_scml2020.html

OneShot SCM Framework

- Agents each control a factory in the supply chain.
- Each day in the SCM world, the following occurs:
  - Exogenous contracts are distributed
  - Negotiations commence
    - Negotiation issues:
      - Price
      - Quantity



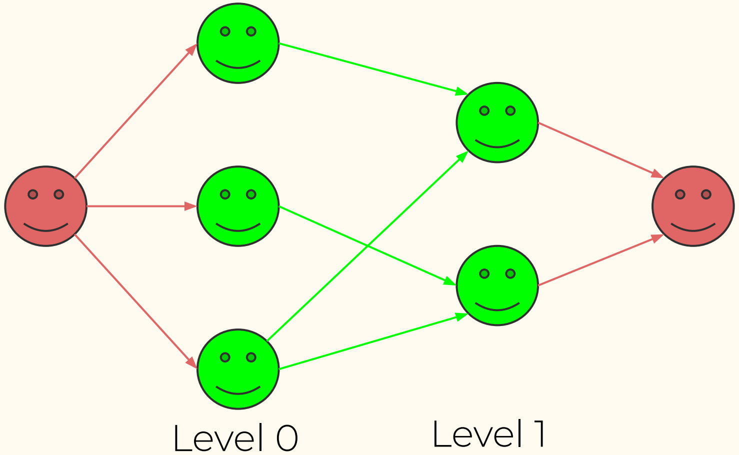
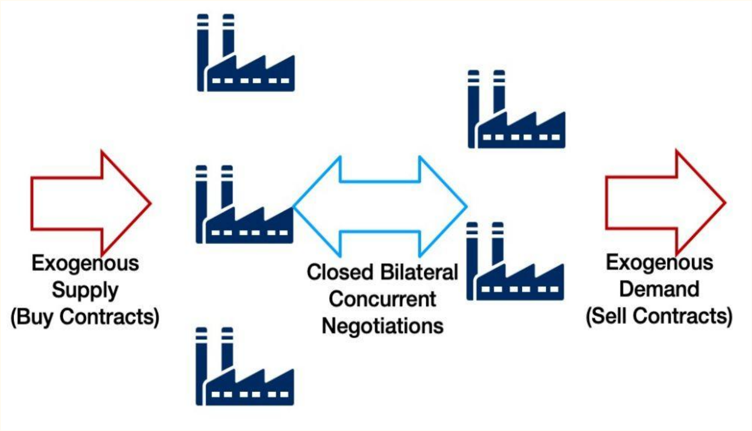


Image source: http://www.yasserm.com/scml/scml2020docs/tutorials/01.run_scml2020.html

OneShot SCM Framework

- Agents each control a factory in the supply chain.
- Each day in the SCM world, the following occurs:
  - Exogenous contracts are distributed
  - Negotiations commence
    - Negotiation issues:
      - Price
      - Quantity
- After X simulation days, agent with highest percent profit wins



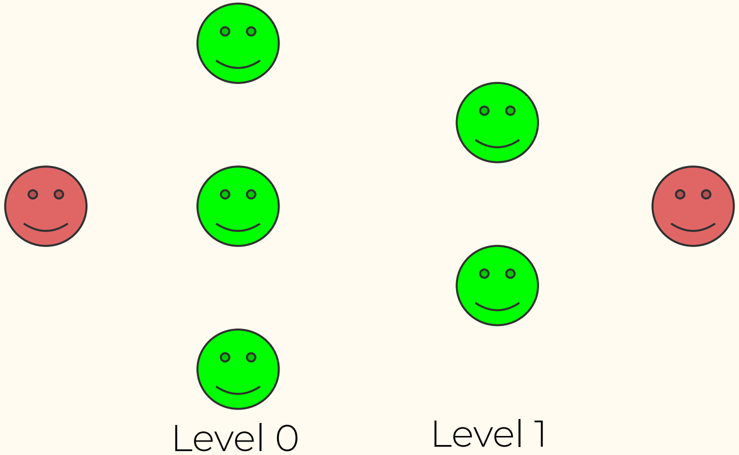
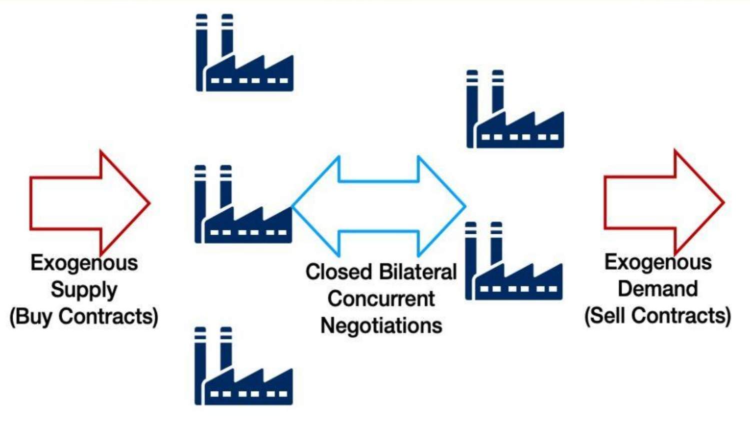


Image source: http://www.yasserm.com/scml/scml2020docs/tutorials/01.run_scml2020.html

OneShot SCM Framework

- Agents each control a factory in the supply chain.
- Each day in the SCM world, the following occurs:
    - Exogenous contracts are distributed
    - Negotiations commence
        - Negotiation issues:
            - Price
            - Quantity
- After X simulation days, agent with highest percent profit wins

- **GOAL:** Design an agent negotiation strategy that performs well in the competition setting



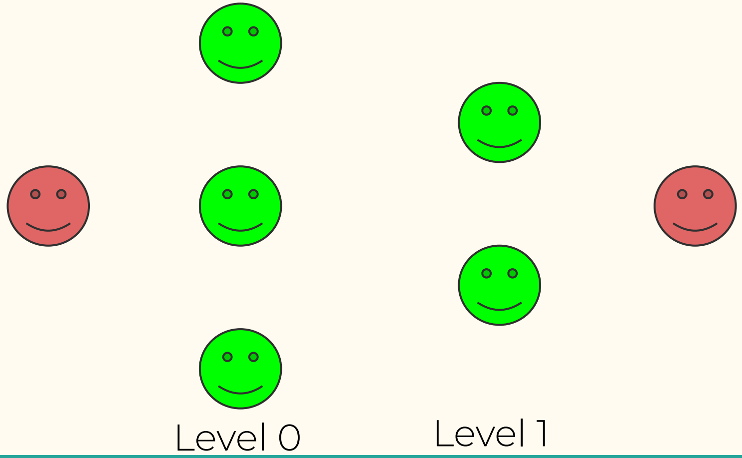


Image source: http://www.yasserm.com/scml/scml2020docs/tutorials/01.run_scml2020.html

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents
    a. Opponent strategies
    b. Opponent exogenous contracts
    c. All negotiation details not involving our agent

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents
   a. Opponent strategies
   b. Opponent exogenous contracts
   c. All negotiation details not involving our agent
2. Many game characteristics are dynamic

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents
    a. Opponent strategies
    b. Opponent exogenous contracts
    c. All negotiation details not involving our agent
2. Many game characteristics are dynamic
    a. Trading prices
    b. Exogenous contracts
    c. Opponent strategies in some cases

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most <u>world information is hidden</u> to varying extents
    a. Opponent strategies
    b. Opponent exogenous contracts
    c. All negotiation details not involving our agent
2. Many <u>game characteristics are dynamic</u>
    a. Trading prices
    b. Exogenous contracts
    c. Opponent strategies in some cases
3. Attempts at developing <u>agents without ML had mixed performance</u>

Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents
   a. Opponent strategies
   b. Opponent exogenous contracts
   c. All negotiation details not involving our agent
2. Many game characteristics are dynamic
   a. Trading prices
   b. Exogenous contracts
   c. Opponent strategies in some cases
3. Attempts at developing agents without ML had mixed performance
   a. At worst, these agents were terrible
   b. At best, these agents were good, but highly sensitive to the strategy environment they played in

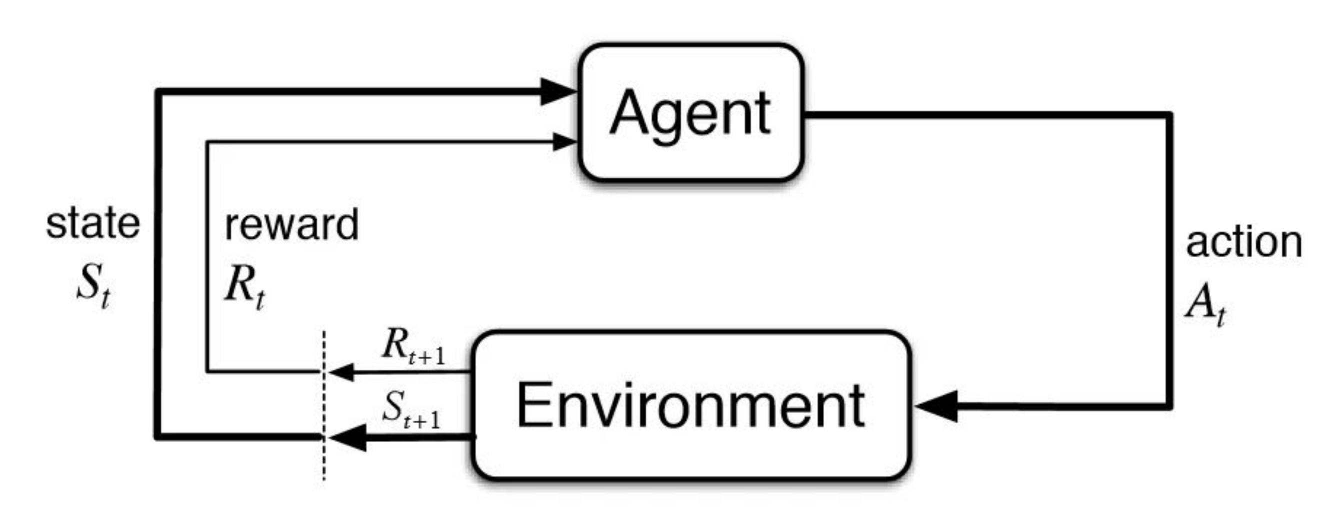
Difficulties

GOAL: Design an agent negotiation strategy that performs well in the competition setting

1. Most world information is hidden to varying extents
2. Many game characteristics are dynamic
3. Attempts at developing agents without ML had mixed performance

FIX: Use reinforcement learning to learn how to play against a variety of opposing negotiation strategies and world environments.

Reinforcement Learning (RL)



Typical Reinforcement Learning cycle

Image source: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Reinforcement Learning (RL)

Some key terms that describe the basic elements of an RL are:

- Agent — A decision-making and action subject.
- Environment — Physical world in which the agent operates
- State — Current situation of the agent
- Action — The behavior of the agent
- Reward — Feedback from the environment
- Policy — Method to map agent’s state to actions

Why use RL?

Features :

- Information (states, actions, rewards) is sent and received between the agent and the environment.
- It uses its own observations (states and rewards) to update its policies, learn better behaviors, and maximize long-term benefits.

Benefits :

- Agents can obtain information about their environment as they act.
- Leads to the discovery of new strategies.
- Effective for parameter optimization.

RL Agent Development

Issues :

- It has drawbacks such as increased training time due to excessive state information and the possibility of overlearning.
- The performance may deteriorate due to insufficient state information.

It's necessary to appropriately select a large amount of state information (e.g., rounds, unit price, quantity, etc.) for agent development in RL.

Algorithms for RL :

- Use Proximal Policy Optimization(PPO) and Q-learning

Proximal Policy Optimization Algorithm

Proximal Policy Optimization Algorithm

- An advantage function A_π: $\mathrm{S} \times \mathrm{A} \rightarrow \mathbb{R}$ is a mapping from state-action pairs to real values according to

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

  where $Q_\pi(s,a)$ is a measure of the quality of a state-action pair and $V_\pi(s)$ is a measure of the average value obtained at state s, all with respect to policy π.

Proximal Policy Optimization Algorithm

- An advantage function A_π: S×A → ℝ is a mapping from state-action pairs to real values according to

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

  where $Q_\pi(s,a)$ is a measure of the quality of a state-action pair and $V_\pi(s)$ is a measure of the average value obtained at state s, all with respect to policy π.

- Algorithm:
  - N agents are given the same policy π_θ to use in parallel environments for T steps

Proximal Policy Optimization Algorithm

- An <u>advantage function</u> A_π: S×A → ℝ is a mapping from state-action pairs to real values according to

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

where $Q_\pi(s,a)$ is a measure of the quality of a state-action pair and $V_\pi(s)$ is a measure of the average value obtained at state s, all with respect to policy π.

- Algorithm:
  - N agents are given the same policy π_θ to use in parallel environments for T steps
  - At each timestep $t \in [1, T]$ and for each agent, approximate an advantage function $\hat{A}_t$

Proximal Policy Optimization Algorithm

- An <u>advantage function</u> A_π: S×A → ℝ is a mapping from state-action pairs to real values according to

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

  where $Q_\pi(s,a)$ is a measure of the quality of a state-action pair and $V_\pi(s)$ is a measure of the average value obtained at state s, all with respect to policy π.

- Algorithm:
  - N agents are given the same policy π_θ to use in parallel environments for T steps
  - At each timestep $t \in [1, T]$ and for each agent, approximate an advantage function $\hat{A}_t$
  - Optimize loss function with respect to policy parameters θ using the advantage function set data

Proximal Policy Optimization Algorithm

- Algorithm:
  - N agents are given the same policy π_θ to use in parallel environments for T steps
  - At each timestep $t \in [1, T]$ and for each agent, approximate an advantage function $\hat{A}_t$
  - Optimize loss function with respect to policy parameters θ using the advantage function set data
  - <u>Main difference</u> between PPO and other policy gradient methods is the loss function:

$$L^{CLIP} = \mathbb{E}_t[\mathrm{clip}(r_t(\theta), 1—\varepsilon, 1+\varepsilon) \cdot \hat{A}_t]$$

where $r_t(\theta)$ measures the "size" of the update to the policy π_{θ}.

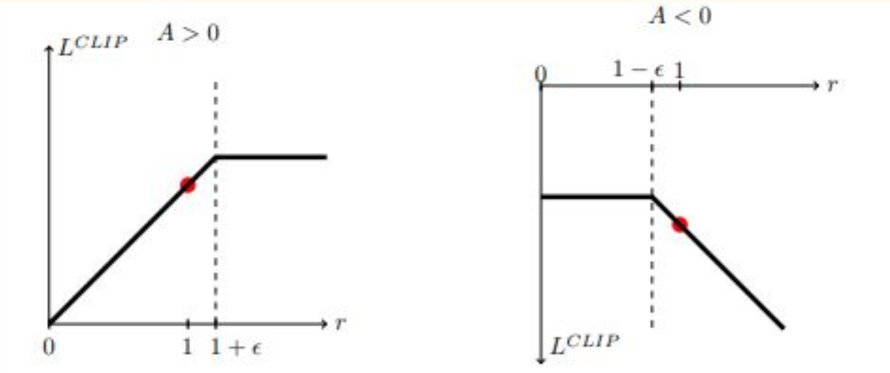


Image Source: https://arxiv.org/pdf/1707.06347.pdf

SCML and RL

The most recent version of SCML implements an interface for using reinforcement learning to train agents

Some aspects of the interface can (and should) be customized to improve performance

Utilizing PPO - Deciding the Observation Space

During the course of an SCML world, a lot of information is available to agents

Need to decide what is useful for the agent to observe:

- Tradeoff: performance vs training time
- Possibility for overfitting to low-relevance information

Can also incorporate negotiation history into observations

Utilizing PPO - Reward Shaping

Another approach: customize the reward given to the agent to encourage “good” behavior

Default reward is based on profit

- Profit is only received at the end of a day
- Profit is dependent on random world variables

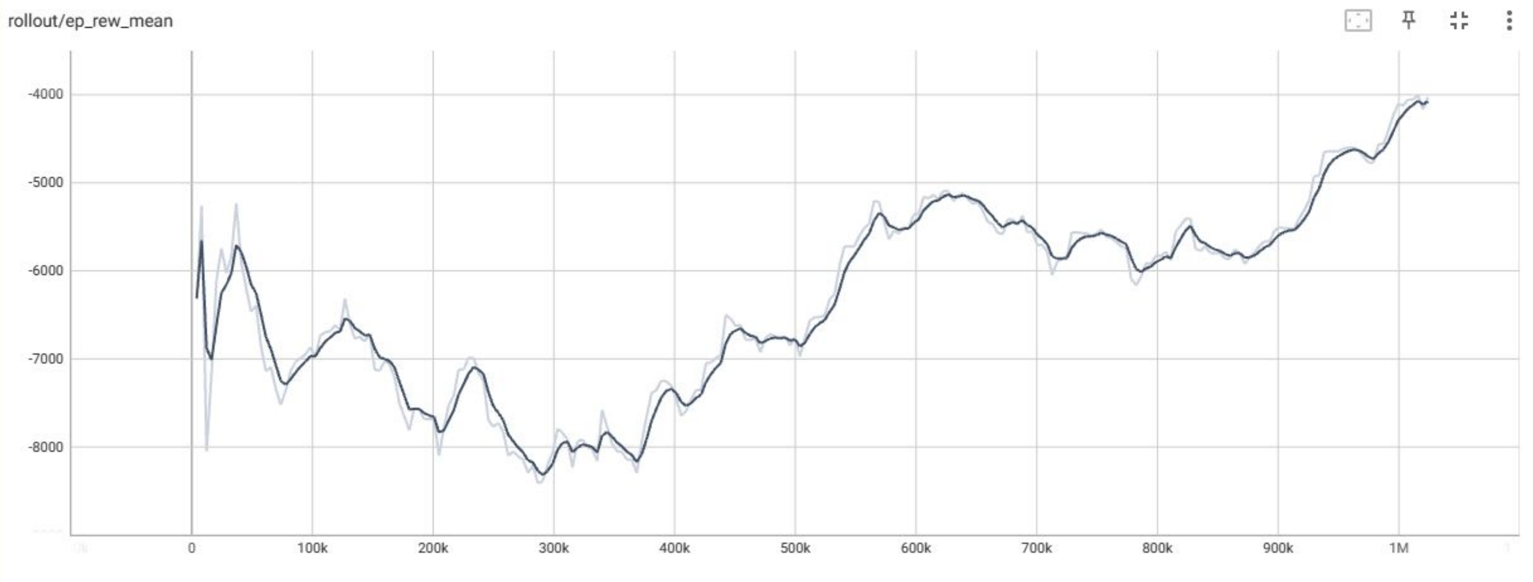
One possible solution: reward the agent for reducing its needs

Utilizing PPO - Training Environment

Default training environment places the agent in a world populated by “default” agents that are present in tournament simulations

Can also train in environments that include other custom agents – this simulates tournament conditions and allows training against agents with known good performance

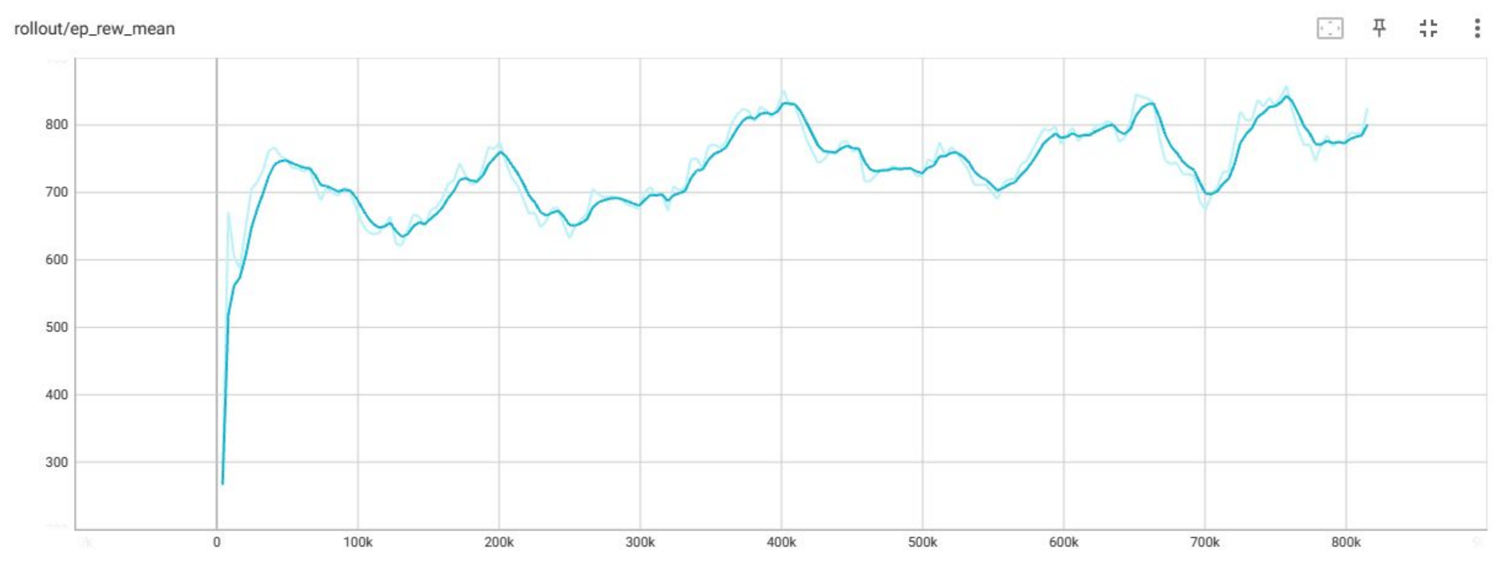
PPO Agents Results – reward shaping




```
Model PPO_L0_4-partners_20230803-144720_1024000-steps_ReducingNeedsReward scores:
[('SyncRandomOneShotAgent', 0.9865932264467818),
 ('SingleAgreementAspirationAgent', 0.8889543832704154),
 ('OneShotRLAgent', 0.7801311137155078),
 ('GreedyOneShotAgent', 0.6383094013967752)]
```

An agent whose reward primarily depends on profit

PPO Agents Results – reward shaping




```
Model PPO_L0_4-partners_20230803-145313_716800-steps_QuantityBasedReward scores:
[('SyncRandomOneShotAgent', 0.9671532567289708),
 ('SingleAgreementAspirationAgent', 0.8820922716538375),
 ('OneShotRLAgent', 0.7320648120499951),
 ('GreedyOneShotAgent', 0.6513504833039025)]
```

An agent whose reward only depends on quantity

Additional work – training & testing scripts

Initial results indicate that performance is still poor – much more experimentation and iteration is needed

To facilitate this, we have developed scripts for easy testing and training

Q-learning method (1/3) - reduce information

In this game, information space is massive

- The number of choices when making proposals:

  $$(2 \times 10)^{\text{number of partners}} \rightarrow 2 \times 10 \text{ or less}$$

  Solution: Use same proposal for all partners (avoiding bad actions very important!)

- Opponent offer information received which we use for reinforcement learning

  $$(2 \times 10)^{\text{number of partners}} \rightarrow 0$$

  Solution: Use a fixed acceptance strategy

Q-learning method (2/3) - how fixed acceptance strategy?

How fixed acceptance strategy?

- In level 0 or step ≥ 19

  $\rightarrow$ the best price combination of offers which the most fulfill needs

- In level 1 and step < 19

  $\rightarrow$ the most offers fulfilling needs which have the best price

And, if remaining needs and partners, propose counter offer. Else, end negotiation.

Q-learning method (3/3) - treat state information -

So, we should estimate only propose unit price and propose quantity

Selected state information for learning

= current step, current needs, level, current number of negotiating partners

Relative to propose unit price

Relative to propose quantity

Number of combination of selected state information for learning = 3360

My method of Q-learning (4/4)

Learning environment

n_steps = 50

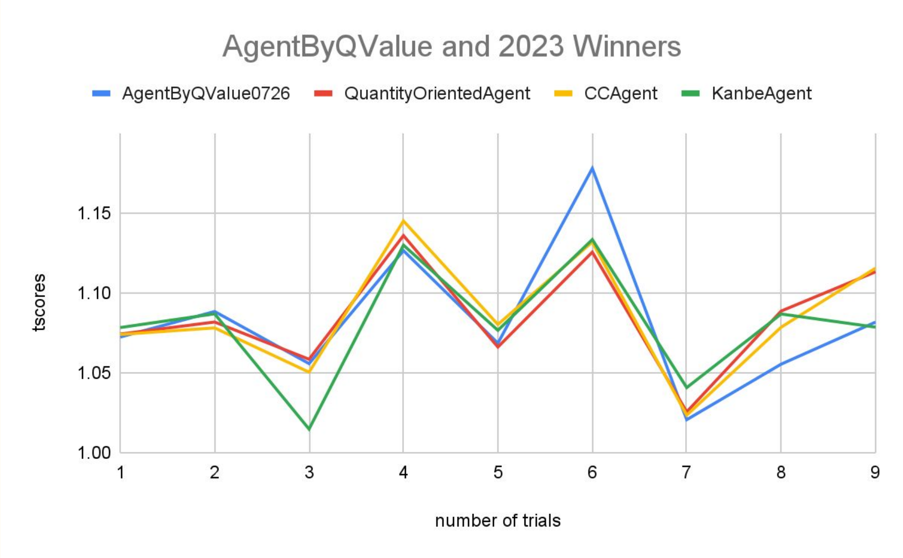
Competitor = [Me, 1st place, 2nd place, 3rd place]

Why Q-learning

- It is easy to make and adjust

All action is random, learning rate = 20, discount factor = 999/1000

Analysis (1/3) - new agent by learning



	mean	median
CCAgent	1.08631	1.07852
QuantityOrientedAgent	1.08546	1.08176
AgentByQValue	1.08299	1.07234
KanbeAgent	1.08064	1.07855

The reason for losing seems the action is not consistent for lack of learning

But, we get strategy that seems best through analysis tendency of best action by learning. So, tried to make new Agent from result of the analysis

Analysis (2/3) - best propose quantity

Average of best propose quantity by learning

		Number of partners					
		1	2	3	4	5	6
needs	1	1	1	1	1	1*	1*
	2	2	1	2	1	1*	1*
	3	3	2	2	2	2	2*
	4	3	3	2	3	2	2*
	5	4	3	3	3	3	3
	6	5	4	4*	4	3	3
	7	5	4	4*	4*	4	3
	8	6*	4*	5*	5	4	4
	9	6*	6*	6*	4	5	6
	10	5*	5*	7*	7	6	6

*: seems to lack of learning Significantly

If number of partners $= 1$

Propose quantity $=$ Needs

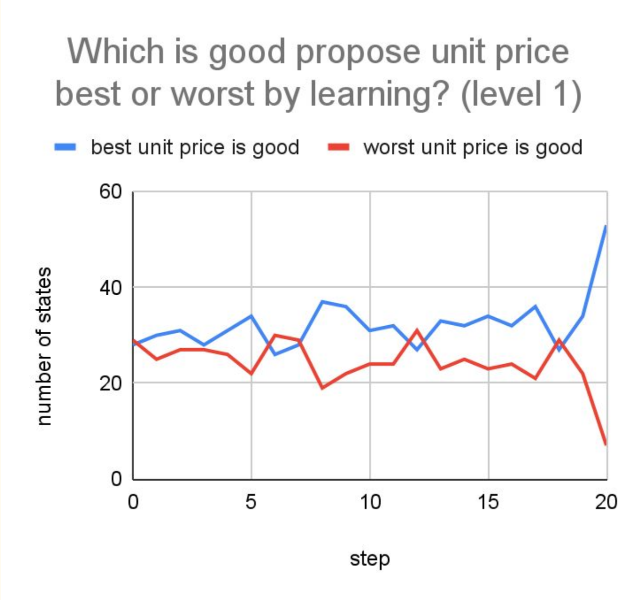
Elif needs ≤ 9

Propose quantity $= \lceil \text{Needs}/2 \rceil$

Else

Propose quantity $= \lceil \text{Needs}/2 \rceil + 1$

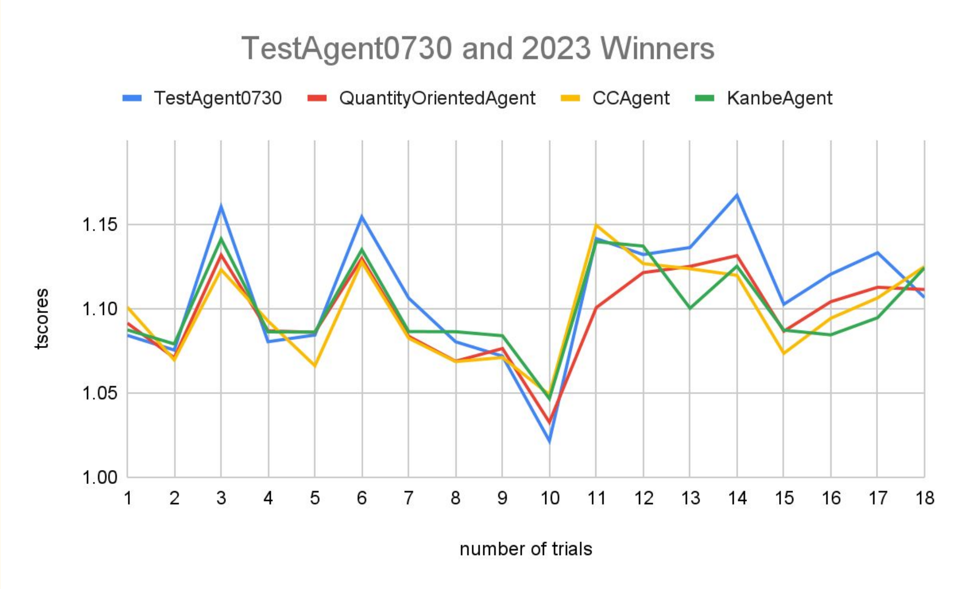
Analysis (3/3) - best propose unit price



Most of the time,
best unit price is better than worst unit price in both level

Propose unit price ＝ best unit price

Result (1/2) - compare with 2023 winners

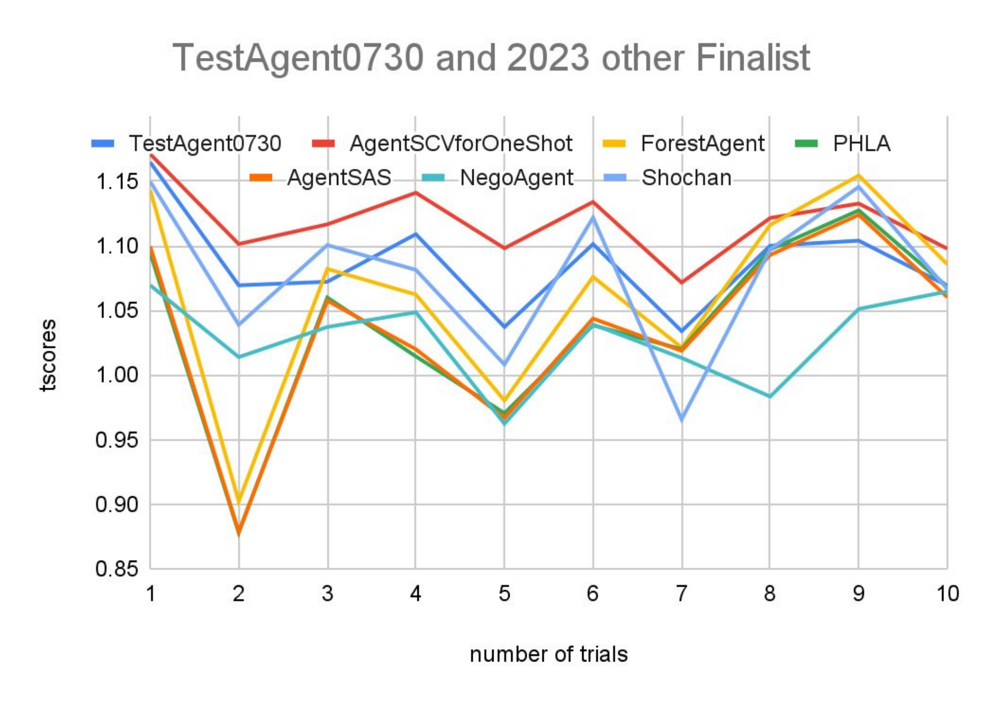


	mean	median
TestAgent0730	1.10879	1.10639
KanbeAgent	1.10058	1.08732
CCAgent	1.09829	1.09755
QuantityOrientedAgent	1.09726	1.09598

Our agent won against agents to be trained on mean and medium (but, our agent lost on minimum)

Against other agents which are not to be trained?

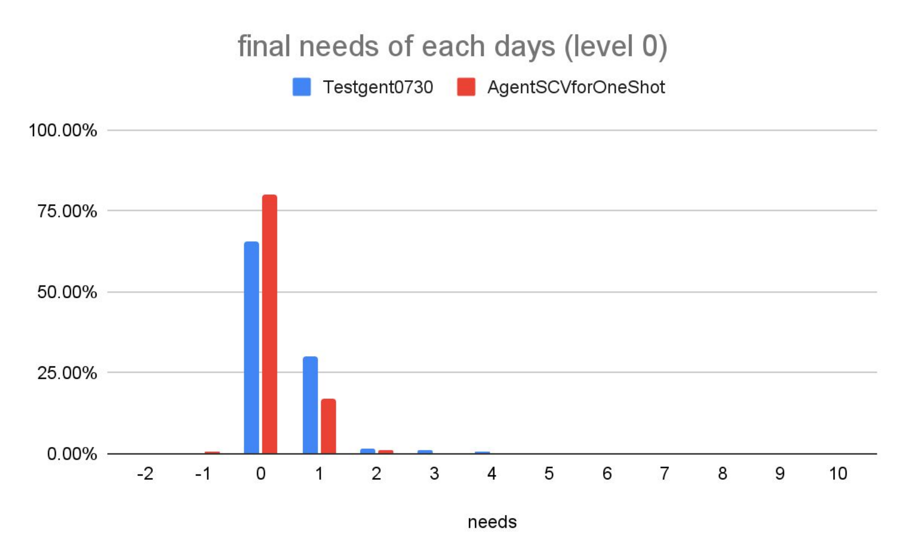
Result (2/2) - compare with 2023 other Finalist

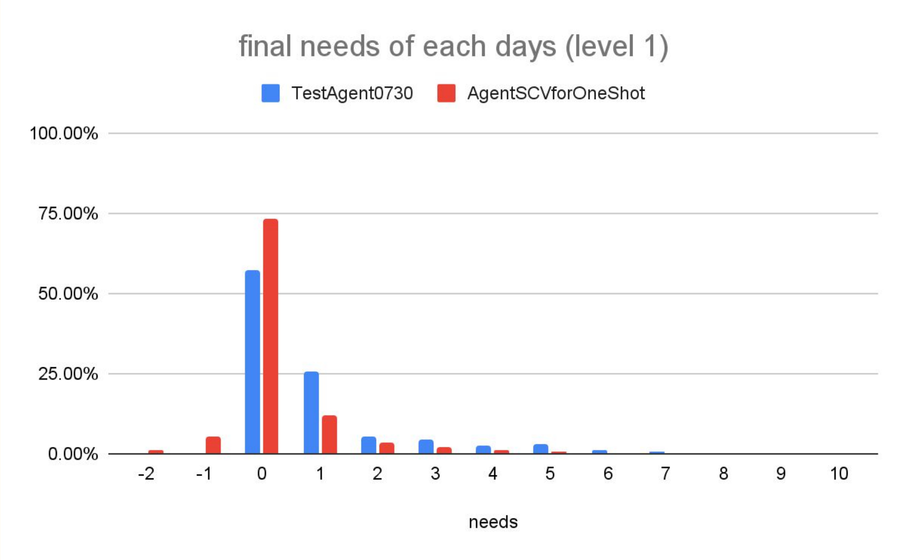


	mean	median
AgentVSCforOneShot	1.11868	1.1192
TestAgent0730	1.08621	1.08625
Shochan	1.07764	1.08939
ForestAgent	1.06246	1.07911
PHLA	1.03695	1.0493
AgentSAS	1.03627	1.05069
NegoAgent	1.02836	1.03822

My Agent lost to specific agent (AgentVSCforOneShot) alltime

Considerations (1/2)- why lost?





Difference between fulfilling needs and not fulfilling needs especially in level 1

Considerations (2/2)

Make new agent TestAgent0803 which have two difference from TestAgent0730 in level 1

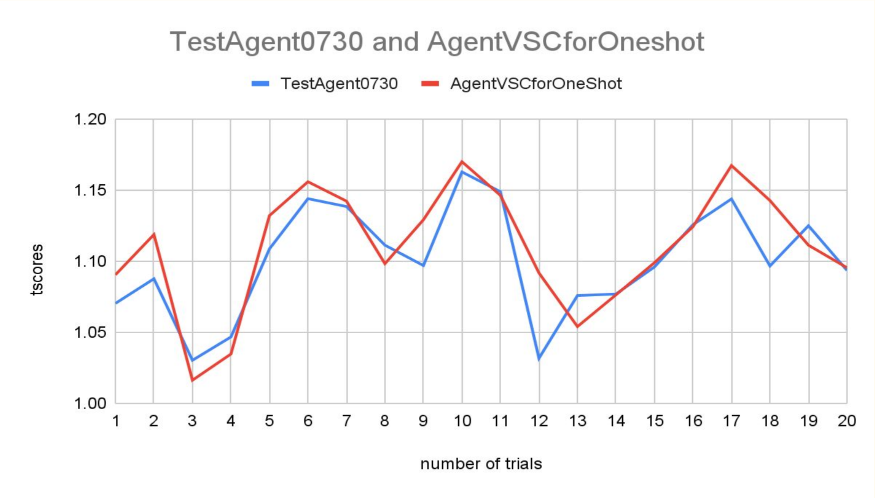
- change the number of steps to completely compromise the unit price from 19 to 18
- compromise the unit price up to permit_needs per day even if the number of steps is less than 18

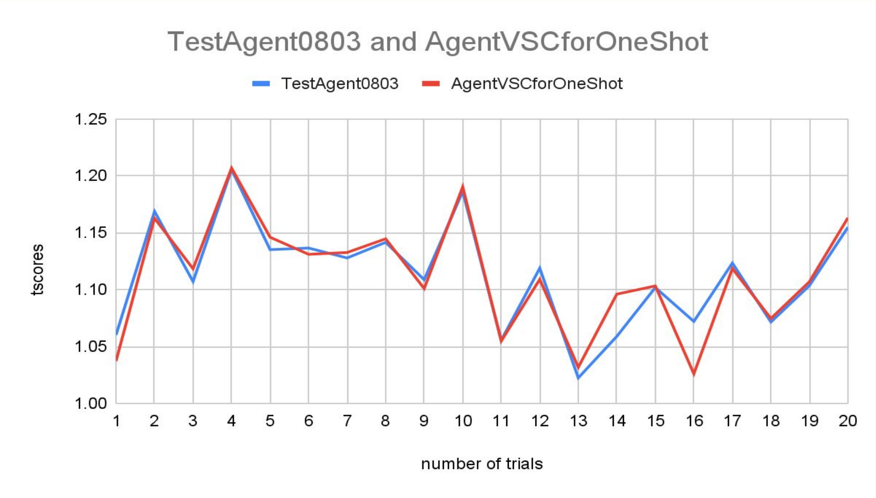
At first, permit needs is 0. And, we renew permit_needs as below every 10 days based on the average of remaining final needs for the last 10 day

$$\text{permit\_needs} \rightarrow \min\{\text{permin\_needs}+1,\ 5\} \quad (\text{the average} > 0.7)$$

$$\text{permit\_needs} \rightarrow \max\{\text{permin\_needs}\ \text{-}1,\ 0\} \quad (\text{the average} < 0.3)$$

New result (1/2) - compare with AgentSCVforOneShot

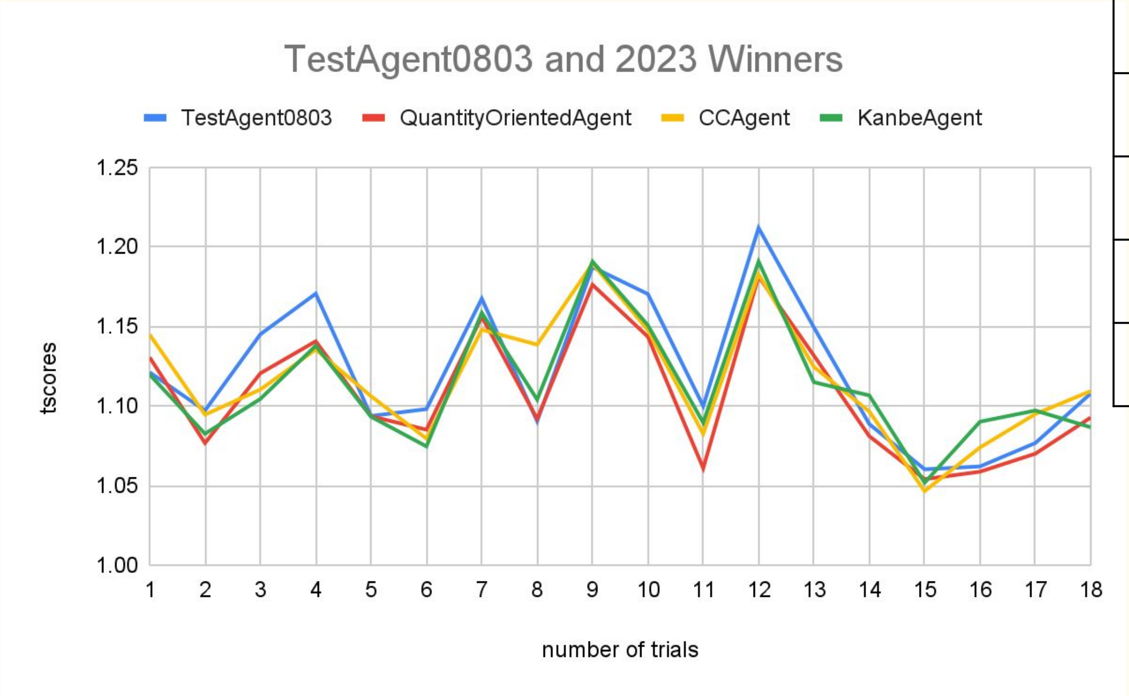




	mean	median
AgentSCVforOneShot	1.1097	1.1149
TestAgent0730	1.1005	1.0967

	mean	median
TestAgent0803	1.1129	1.1137
AgentSCVforOneShot	1.1127	1.1136

New result (2/2) - compare with 2023 winners



	mean	median
TestAgent0803	1,1220	1.1038
CCAgent	1.1169	1.1098
KanbeAgent	1.1135	1.1042
QuantityOrientedAgent	1.1080	1.093

Of course, TestAgent0803 can take good performance against 2023 winners.

Future work (Q-learning)

- Consider level-dependant acceptance strategies that are more adaptable to the environment
- Optimize each parameter
- Use better hardware in order to evaluate and optimize each parameter

Conclusion

Developing agents for SCML is very difficult!

Open-ended RL achieves modest performance – more tuning and iteration may allow it to achieve much better results

Mixed RL and hardcoded strategies achieve good results without needing a great deal of engineering