Negotiation Strategy using Reinforcement Learning for OneShot Track

Takumu Shimizu
Tokyo University of Agriculture and Technology
Katsuhide Fujita Lab.

Contents

RLAgent

- Negotiates with the opponents independently
  - Inherits SimpleAgent
- Applied Reinforcement Learning
  - Uses Proximal Policy Optimization (PPO) algorithm
  - Improves PPO-PyTorch[1]
- Defines the Markov Decision Process (MDP)
  - Adjusts to the opponent's strategies in OneShot Track

[1] Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

MDP for OneShot Track

- **State** consists of the following factors:
  - The current number of rounds $r \in \{0, 1, \ldots, R\}$
    - R is the negotiation deadline
  - The current needs q_r^{needs}
    - The exogenous contract quantity minus the quantity of the contract with other competitors
  - The opponent's offer $\omega_r'^a$
    - Consists of the quantity q', the negotiation time t', and the unit price p'

Possible values of items in the opponent's offer $\omega_r'^a$

Item	Value
quantity q'	0 ~ 10
time t'	0 ~ 200
Unit price p'	High or Low

MDP for OneShot Track

- **Action** consists of the following factors:
  - The accept signal η_r^a
    - Indicates whether to accept or reject an opponent's offer
  - The counter offer ω_r^a
    - Consists of the quantity q, and the unit price p

Possible values of items in the counter offer ω_r^a

Item	Value
quantity q'	0 ~ 10
Unit price p'	High or Low

MDP for OneShot Track

- **Reward** is the profit of the day
  - Calculated by the utility function (OneShotUfun)
    - Using the contract with the competitor and the exogenous contract
  - RLAgent get the profit as the reward in the last round of the day
    - Otherwise, the reward is 0

RL Agent Negotiation Strategy

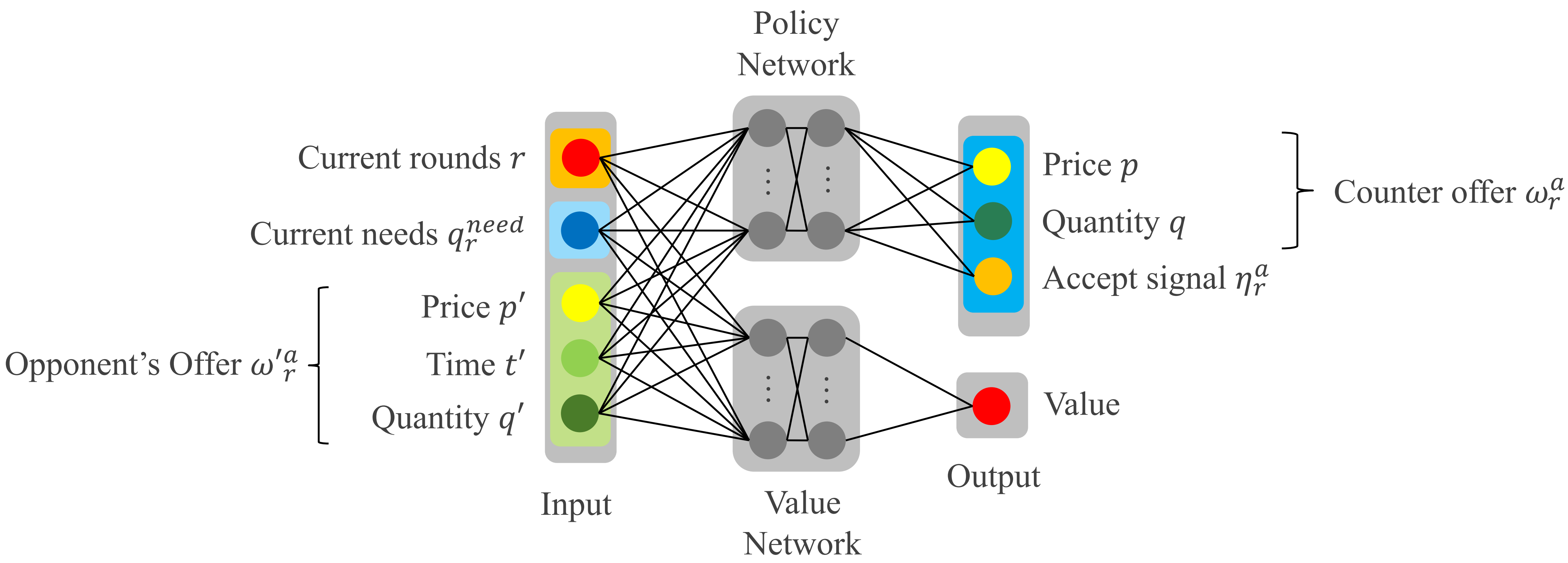
1. Receives the opponent's offer
   - In the first round, it receives the supposed offer
2. Enters the offers into the model as the state and gets an action
3. Sends the response to the opponent
   - Depends on the accept signal η_r^a
     - **True**: an acceptance response
     - **False**: a counter offer ω_r^a
   - When the needs $q_r^{needs} \leq 0$, RLAgent ends the negotiation

How to train RLAgent

- Conducted the simulation of about 1500 worlds to train the model
  - Opponents are the sample agents
    - SimpleAgent
    - AdaptiveAgent
    - LearningAgent
- The best model is used in the evaluation
  - It has the best score in the training phase

Model Overview (RLAgent)

- State and Action are expressed by MultiDiscrete
  - Converts each item to one-hot representation



RLSyncAgent

- Negotiates with the opponents concurrently
  - Inherits SyncAgent
- Applied Reinforcement Learning
  - Uses Proximal Policy Optimization (PPO) algorithm
  - Improves PPO-PyTorch[1]
- Defines the Markov Decision Process (MDP)
  - Adjusts to the opponent's strategies in OneShot Track

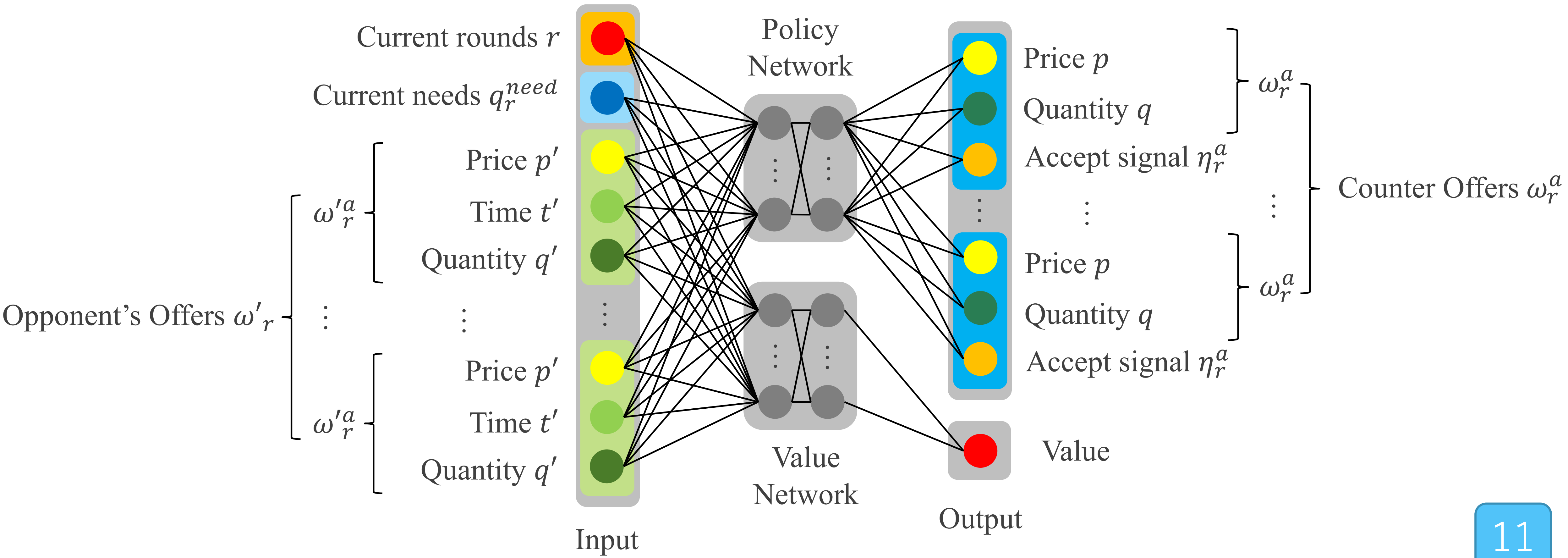
[1] Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

Difference from RLAgent

- Deals with the multiple offers at the same time
- State
  - The opponent's offer: $\omega_r'^a$ $\rightarrow$ The set of the opponent's offers: ω_r'
- Action
  - The counter offer: ω_r^a $\rightarrow$ The set of the counter offers: ω_r
- Model
  - Number of nodes are added with changes in the state and the action

Model Overview (RLSyncAgent)

- The nodes of the input layer and the output layer are added



Evaluation

- Trained agents had a lower utility value than the sample agents
- Submitted **RLAgent** to the competition

Table 1: The test results of RLAgent and RLSyncAgent

Agent	score	min	Q1	median	Q3	max
RLAgent	0.927	0.708	0.864	0.947	0.991	1.051
RLSyncAgent	0.712	0.173	0.461	0.809	0.910	1.056
SimpleAgent	1.035	0.595	1.004	1.080	1.127	1.204
AdaptiveAgent	0.978	0.620	0.883	0.989	1.083	1.206
LearningAgent	0.982	0.618	0.881	0.981	1.110	1.212

Summary

- RLAgent
  - Applied Reinforcement Learning for OneShot Track
  - Negotiates with the opponents independently
- RLSyncAgent
  - Negotiates with the opponents concurrently
  - Deals with multiple offers at the same time
- Evaluation
  - RLAgent gets higher utility than RLSyncAgent

Thank you for listening

appendix

MDP for OneShot Track

- **State** consists of following factors:
  - The current number of rounds $r \in \{0,1,\dots,R\}$
    - R is the negotiation deadline
  - The current needs q_r^{needs}
    - The exogenous contracts quantity minus
      the quantity of the contract with other competitors
  - The opponent's offer ω'_r
    - ω'_r is the set of opponent's offer ω'^{a}_r
    - ω'^{a}_r consists of the quantity q', negotiation time t', and the unit price p'

MDP for OneShot Track

- **Action** consists of following factors:
  - The accept signal η_r^a
    - Indicates whether to accept or reject an opponent's offer
  - The counter offer ω_r
    - ω_r is the set of ω_r^a
    - ω_r^a consists of the quantity q, and the unit price p

MDP for OneShot Track

- **Reward** is the profit of the day
  - Calculated by the utility function (OneShotUfun)
    - Uses the contract with the competitor and the exogenous contract
  - RLAgent gets the profit as reward in the last round of the day
    - Otherwise, the reward is 0

Negotiation Strategy

1. Receives the opponent's offer
   - In the first round, it receives the supposed offer
2. Enters the model as state and get an action
3. Sends the response to the opponent
   - Depends on the accept signal η_r^a
     - **True**: an acceptance response
     - **False**: a counter offer ω_r^a
   - When the needs $q_r^{needs} \leq 0$, RLAgent end the negotiation

How to learn

- Conducted about 1500 simulations to train the model
  - Opponents are the sample agents
    - SimpleAgent
    - AdaptiveAgent
    - LearningAgent
- The best model is used in the evaluation
  - It has the best score in the training phase

Discussion

- RLAgent gets lower score than sample agents
  - RLAgent cannot consider other negotiations
  - It is possible that the environment is too complex to learn well.
- RLSyncAgent gets significantly lower on all scores
  - It works fine regarding the acceptance of the offer.
    - It can adjust the total quantity of the contracts to the proper value
  - The challenge is how to make the offer
    - it is difficult to adjust the total quantity due to predictions of accepted offers