Deep Reinforcement Learning Framework with Representation Learning for Concurrent Negotiation

Ryoga Miyajima and Katsuhide Fujita
Tokyo University of Agriculture and Technology, Koganei, Tokyo, Japan

Keywords: Automated Negotiation, Concurrent Negotiation, Multi-Agent System, Supply Chain Management, Reinforcement Learning, Representation Learning.

Abstract: In the field of automated negotiation, significant attention has been paid to methods for learning negotiation strategies using reinforcement learning. However, in concurrent negotiation, where negotiation proceeds with multiple counterparties with various strategies in parallel, it is difficult to consider the differences in the strategies of counterparties using the conventional formulation in which the state is defined using the bids of both counterparties. In this study, we propose a reinforcement learning framework for learning negotiation strategies that considers the strategy models of the negotiation partners in concurrent negotiations. Strategy modeling is realized using embeddings with a representation function based on the unsupervised learning of generative–discriminative representations from negotiation log data. Through evaluation experiments, we show the performance of the representation function in identifying the strategies of negotiation partners and the effectiveness of introducing the representation function into the reinforcement learning of negotiation strategies.

1 INTRODUCTION

In the real world, agents in multi agent systems (MAS) are typically not under integrated control for various reasons, including different owners and difficulty in centralized control. Therefore, when each agent acts according to its preferences, conflicts may occur. In such cases, automated negotiation is attracting attention as a technology for resolving conflicts and reaching agreements. One of the applications of automated negotiation is supply chain management (SCM). In SCM, it is necessary to reach an agreement with the factories that are counterparties to the supply chain (SC) on contract terms for incoming and outgoing products. When an agent oversees the negotiations with each factory, it is a concurrent negotiation problem in which negotiations with multiple agents are conducted in parallel, because there are typically multiple candidate factories with which to do business.

Recently, a method for learning negotiation strategies using reinforcement learning (RL) has been proposed for bilateral negotiation, but few for concurrent negotiation. In concurrent negotiation, opponent agents typically have different strategies. Therefore, it is necessary to learn the strategies of negotiation partners to reach an agreement with each of them so that individual utility can be fully obtained in the negotiation.

In this study, we propose an RL agent that takes appropriate negotiation strategies according to the negotiation strategies of each of the multiple negotiating agents in automated negotiation for transactions in SCM. When introducing RL into concurrent negotiation, it is difficult to consider the differences in the strategies of the counterparties in the conventional formulation. Therefore, we propose an RL framework that introduces a modeling method of the strategies of negotiation partners by representation learning and demonstrate the effectiveness through experiments. Experiments are conducted in environments with various combinations of negotiation partners and evaluate the effectiveness of each of them and the robustness of the representation learning.

2 RELATED WORK

2.1 Automated Negotiation

Automated negotiation has attracted attention as a method to resolve conflicts by reaching agreements

Miyajima, R. and Fujita, K.
Deep Reinforcement Learning Framework with Representation Learning for Concurrent Negotiation.
DOI: 10.5220/0012336000003636

In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 1*, pages 231-239
ISBN: 978-989-758-680-4; ISSN: 2184-433X

through negotiations between agents. The Automated Negotiating Agent Competition (ANAC) is an international tournament to compete in the performance of negotiating agents (Baarslag et al., 2015). ANAC has various leagues with different scenarios, one of them is Supply Chain Management League (SCML) (Mohammad et al., 2019). SCML is the only league that handles concurrent negotiation problems. In SCML, negotiations are performed with factories in neighboring layers of the SC on the unit price, quantity, and delivery date of incoming and outgoing products. One of the tracks held at SCML2022 was the OneShot Track (Mohammad et al., 2022). In OneShot Track, the SC has two layers and each factory cannot carry over inventory to the next day or later. Thus, unsold inventory is discarded at the end of the day. The agent serves one factory in the SC and negotiates with the agents in the other layer on the unit price and quantity of the products to be traded. OneShot Track is a more focused negotiation-only track than the other tracks because it does not need to manage inventory across days.

In recent years, significant attention has been paid to methods for acquiring negotiation strategies through RL. Bakker et al. proposed the RLBOA framework as a framework that enables the learning of bidding strategies by RL in bilateral negotiations (Bakker et al., 2019). Bagga et al. proposed ANEGMA, an RL model for concurrent negotiation in e-markets, which was efficient for negotiation partners with a common strategy (Bakker et al., 2019).

2.2 Representation Learning

Grover et al. proposed a generic learning framework for modeling agent behavior in MAS (Grover et al., 2018). The method formulated MAS as an extension of the Partially Observed Markov Decision Process (POMDP) and learned unsupervised a representation function that embedded interaction episodes with other agents in a real-valued vector for the objective of generative–discriminative representation.

The representation function $f_\theta : E \to \mathbb{R}^d$ is a function that embeds an interaction episode $e \in E$ with a particular agent in the MAS in a real-valued vector. It is learned unsupervised using past interaction episodes for the following generative–discriminative representations.

- Generative representations: Useful representations for simulating the policy of an agent
- Discriminative representations: Representations that distinguish the policy of an agent from the policies of other agents

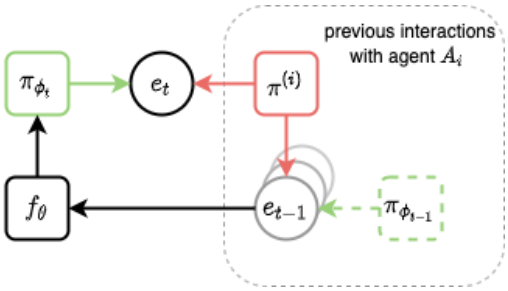

Figure 1: The learning framework using representation function (Grover et al., 2018).

Generative–discriminative representations are learned by maximizing the objective function J, which is computed using Equation (1).

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\substack{e_+\sim E_i,\\ e_*\sim E_i\setminus e_+}} \left[\underbrace{\sum_{\langle o,a\rangle\sim e_+} \log \pi(a|o, f_\theta(e_*))}_{\text{imitation}} - \underbrace{\lambda \sum_{j\neq i} \mathbb{E}_{e_-\sim E_j}[d_\theta(e_+, e_-, e_*)]}_{\text{agent identification}} \right],$$

$$d_\theta(e_+, e_-, e_*) = (1 + \exp(||f_\theta(e_*) - f_\theta(e_-)||_2 - ||f_\theta(e_*) - f_\theta(e_+)||_2))^{-2}, \quad (1)$$

where n denotes the number of the other agents, and E_i denotes the set of interaction episodes with A_i. An interaction episode $e \in E_i$ is a pair (o,a) of observations and actions in the interaction with A_i.

The learning framework proposed by Grover et al. that introduces a representational function is depicted in Figure 1. The interaction episode e_{t-1} between the learning agent and agent A_i when the learning agent acts on the policy $\pi_{\phi_{t-1}}$ and A_i acts on the policy $\pi^{(i)}$ is embedded in a real-valued vector by the representation function f_θ. The policy conditioned by this vector determines the behavior.

3 CONCURRENT NEGOTIATION PROBLEM IN SCM

3.1 Negotiation Environment in SCM

This study deals with the concurrent negotiation problem shown in Figure 2, which is a partially modified environment used in SCML2022 OneShot Track (Mohammad et al., 2022) implemented with NegMAS (Mohammad et al., 2021). An agent is responsible for one factory in a two-layer SC consisting of L_0 and L_1 and negotiates with agents in the layer. For simplicity, this study assumes that there is only one agent.

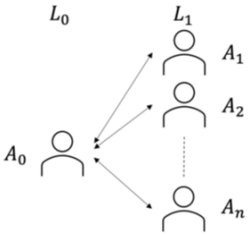

Figure 2: Concurrent negotiation problem handled in this study.

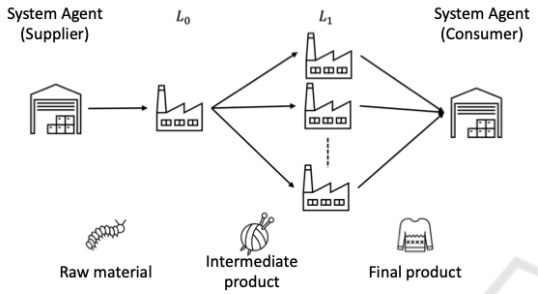

Figure 3: Flow of products in the SC assumed in this study.

We denote the agent in L_0 as A_0 and the agents in L_1 as $A_1, A_2, \ldots, A_n$.

The flow of products in the SC assumed in this study is depicted in Figure 3. Production and transactions occur once a day. At the beginning of each day, all agents receive exogenous contracts from the system agents, which specify the unit price and quantity of the raw materials the agent in $L0$ buys and the final products the agents in $L1$ sell. Then, agents in $L0$ and $L1$ negotiate the unit price and quantity of the intermediate product traded between them. After all negotiations are completed, the contracts are executed and the products are produced at production cost. A disposal cost is incurred for remaining materials according to their trading value, and a penalty is imposed for shortages against contracts according to their trading value.

3.2 Negotiation Protocol

In this study, we use the alternating offers protocol (AOP) (Rubinstein, 1982).In bilateral negotiations using the AOP, the two agents take turns proposing bids. When one agent proposes to the other agent, the proposed agent selects one of the following three actions.

- Accept: Accept the bid proposed by the other agent.
- Offer: Reject the bid proposed by the other agent, and propose its bid.
- EndNegotiation: Terminate negotiation without reaching an agreement.

This is repeated until the negotiation is terminated by Accept or EndNegotiation or until the number of rounds is reached. In concurrent negotiation, an agent proposes a bid to each of its counterpart agents and selects an action to respond to each bid received from its counterpart agents.

3.3 Utility Function

The utility function is a function that outputs the utility value obtained by agreements. In this study, for simplicity, all agents are assumed to determine their utility values by the unit price $v^p_{c_p}$ only. This means that the utility value $U(\omega)$ of a single bid $\omega = (v^p_{c_p}, v^q_{c_q})$ for A_0 is defined using Equation (2), and that for $A_1, A_2, \ldots, A_n$ is defined using Equation (3).

$$U(\omega) = \frac{v^p_{c_p} - v^p_{min}}{v^p_{max} - v^p_{min}}, \tag{2}$$

$$U(\omega) = \frac{v^p_{max} - v^p_{c_p}}{v^p_{max} - v^p_{min}}, \tag{3}$$

where $v^p_{min} \leq v^p_{c_p} \leq v^p_{max}$.

4 DEEP REINFORCEMENT LEARNING FOR CONCURRENT NEGOTIATION

4.1 Formulation of Concurrent Negotiation Problem

We formulate the concurrent negotiation problem as a Markov Decision Process (MDP) by focusing on negotiations between individual opponent agents. We formulate the negotiation with agent A_i as $MDP_i = \langle \mathcal{S}_i, \mathcal{A}_i, r^i, \mathcal{T} \rangle$. We describe the agent's state, action, and reward.

State $s^I \in \mathcal{S}_i$

The state consists of the following four elements.

- Elapsed days normalized to $[0,1]$, $d/(D-1)$.
- The negotiation round normalized to $[0,1]$, $r/(R-1)$.
- The utility value of the bid that the learning agent proposed to A_i at the previous round, $U(\omega^{\text{for } i}_{r-1})$.
- The utility value of the bid that A_i proposed to the learning agent just before, $U(\omega^{\text{from } i}_{r-1})$.

Action $a^I \in \mathcal{A}$

The action is the target utility of the bid next proposed to agent A_i, $u^{\text{for } i}_{target}$.

Reward $R^i : \mathcal{S}_i \times \mathcal{A}_i \to \mathbb{R}$

Rewards are granted for each negotiation round with each counterparty. These rewards are divided into three categories: agreement reaching, negotiation failure, and continuation of negotiation.

- Reaching an agreement: The utility value of the agreed bid ω, $r^i = U(\omega)$.
- Negotiation failure: $r^i = -0.5$
- Continuation of negotiation: Rewards are given in the following cases according to the change in the utility value of the bid of the opponent in the next round.
  - $U(\omega^{\text{from } i}_{r-1}) = U(\omega^{\text{from } i}_{r})$: $r^i = 0$
  - $U(\omega^{\text{from } i}_{r-1}) > U(\omega^{\text{from } i}_{r})$: $r^i = -0.01$
  - $U(\omega^{\text{from } i}_{r-1}) < U(\omega^{\text{from } i}_{r})$: $r^i = 0.01$

4.2 Representation Function

We introduce a representation function to model the strategies of the negotiating partners. The representation function $f_\theta : E \to \mathbb{R}^d$ is a function that embeds the interaction episode $e \in E$ with a particular agent in a real-valued vector, as described in Section 2.2. In this study, because the negotiation problem is formulated as an MDP with each negotiation partner as described in Section 4.1, and the state consists of its proposal and that of the opponent, we consider the state the interaction episode and use as the input to the representation function. In learning the representation function, the parameter θ is updated to maximize the objective function $J(\theta)$ obtained using Equation (4).

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\substack{e_+ \sim E_i, \\ e_* \sim E_i \setminus e_+}} \left[\sum_{\langle s,a \rangle \sim e_+} \log \pi(a|s, f_\theta(e_*)) - \lambda \sum_{j \neq i} \mathbb{E}_{e_- \sim E_j}[d_\theta(e_+, e_-, e_*)] \right],$$
$$d_\theta(e_+, e_-, e_*) = (1 + \exp(||f_\theta(e_*) - f_\theta(e_-)||_2 - ||f_\theta(e_*) - f_\theta(e_+)||_2))^{-2}, \quad (4)$$

where $e \in E_i$ denotes the interaction episodes in one day of the simulation, which are the time series data up to the end of the negotiation of the state, action pairs in the negotiation with agent A_i. Furthermore, the output of the representation function for the input $e \in E_i$ is obtained by averaging the output of the representation function in each state as $f_\theta(e) = \frac{1}{n_e}\sum_{\langle s,a \rangle \sim e} f_\theta(s)$, where n_e denotes the length of e.

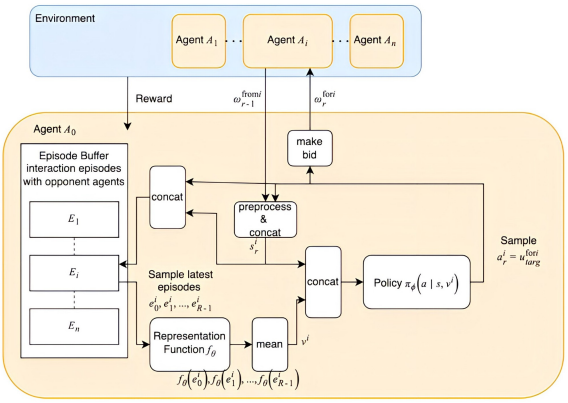

Figure 4: The proposed learning framework for concurrent negotiation.

4.3 Learning Framework

The learning framework is depicted in Figure 4. When the learning agent A_0 is proposed a bid $\omega^{\text{from } i}_{r-1}$ by agent A_i, it transits to state s^i_r. This state s^i_r is added to buffer E_i to hold the interaction episode with A_i in concatenation with action a^i_r that is subsequently determined. The real-valued vector v^i that models the strategy of A_i is obtained by averaging the output of the representation function f_θ for the interaction episodes $e^i_0, e^i_1, \ldots, e^i_{R-1}$, which are the most recent interaction episodes with A_i and stored in E_i. Then, s^i_r and v^i are concatenated and input to the stochastic policy π_ϕ to select the action a^i_r. The action a^i_r selected here is the target utility $u^{\text{for } i}_{target}$ to make a bid $\omega^{\text{for } i}_r$. The reward is given by the response of A_i to the proposal $\omega^{\text{for } i}_r$.

The policy π_ϕ is updated by storing the states, actions, and rewards in a buffer and calculating the loss from these stored data with all opponents at the end of each day. Moreover, to update the representation function f_θ, it samples $e_+, e_* \sim E_i, e_- \sim E_j$ from the interaction episode buffer $E_i, E_j (1 \leq i, j \leq n, i \neq j)$ and then update θ using the gradient of $J(\theta)$, which is obtained using Equation (4). This is performed for all (A_i, A_j) permutations of the two opponent agents. The policy and representation function learn in parallel during repeated simulations.

4.4 Learning Architecture

In this study, RL is performed using the actor–critic because of its learning efficiency by simultaneously improving the policy and estimating the value (Konda and Tsitsiklis, 1999). The architecture is shown in Figure 5 as the model diagram of the actor, Figure 6 as that of the critic, and Figure 7 as that of the representation function. Both inputs of the actor and the critic

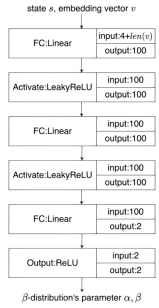

Figure 5: Model diagram of the actor.

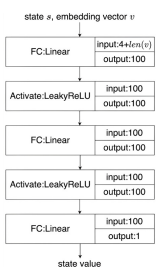

Figure 6: Model diagram of the critic.

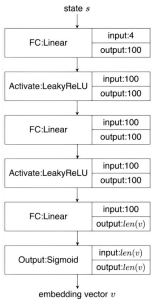

Figure 7: Model diagram of the representation function.

are a concatenated vector of the state s and embedding vector v. The output of the actor is the parameters of the stochastic policy π_ϕ with beta distribution, and the output of the critic is the state value. The input of the representation function is the state s, and the output is the real-valued embedding vector.

5 EVALUATION EXPERIMENTS

5.1 Negotiation Environment Settings

The negotiation environment in this experiment is the setting described in Section 3. That is, the learning agent A_0 is responsible for the L_0 factory in the SC shown in Figure 3 and negotiates with the agents $A_1, A_2, \ldots, A_n$ responsible for the L_1 factories.

Configuration Parameters

The production costs for A_i is $p^i_{produce}$, which is sampled by $U(1,10)$ if $i = 0$ otherwise $2U(1,10)$ where $U(a,b)$ is the uniform distribution over $[a,b]$. The shortfall penalty coefficient is 0.6, and the disposal cost coefficient is 0.1. The quantity of raw materials supplied by the exogenous contract per day is 10, and the total quantity of final products demanded by the exogenous contracts per day is also 10.

Opponent Agents

We use the four types of agents for L_1 factories. These agents differ only in how to calculate their target utility values u^{target}. They propose a bid whose unit price corresponds to u^{target} according to Equation (3) and whose quantity is the smaller of their required quantity based on the exogenous contract and their agreed contracts and the quantity proposed by the opponent agent just before. They only accept a bid where its utility value is not lower than u^{target} of their next proposal and its quantity is not more than their required quantity. In addition, they terminate negotiation when the total quantity of intermediate products contracted through negotiations exceeds the quantity of final products demanded under the exogenous contract. We describe below how to calculate u^{target} for each agent.

- **TimeDependentAgent(TiD)**
  TiD concedes the target utility value ranging from 1 to 0 over time. The target utility value for the rth round $(0 \leq r < R)$ of the TiD u_r^{target} is determined using Equation (5).

$$u_r^{target} = \left(1 - \frac{r}{R-1}\right)^e \tag{5}$$

  The parameter e determines the shape of the concession curve, and we adopt $e = 0.2$. The following e parameters for CTD and Ada are also 0.2.

- **CrampedTimeDependentAgent(CTD)**
  CTD concedes the target utility value ranging from 1 to $\alpha(0 < \alpha < 1)$ over time. The target utility value for the rth round of the CTD u_r^{target} is determined using Equation (6).

$$u_r^{target} = \alpha + (1-\alpha)\left(1 - \frac{r}{R-1}\right)^e \tag{6}$$

  In this experiment, we adopt $\alpha = 0.2$.

- **TitForTatAgent(TFT)**
  TFT determines the target utility value depending on the opponent's behavior. If the opponent concedes, it also concedes, and if the opponent becomes bullish, it also becomes bullish. The target utility value for the rth round of the TFT u_r^{target} is determined using Equation (7).

$$u_r^{target} = \begin{cases} 1 & \text{if } r = 0,1 \\ \frac{U(\omega_{r-2}^{\text{from } i}) + \varepsilon}{U(\omega_{r-1}^{\text{from } i}) + \varepsilon} U(\omega_{r-1}^{\text{for } i}) & \text{otherwise} \end{cases} \tag{7}$$

  The ε is a term to prevent zero division and abrupt changes. In this experiment, we adopt $\varepsilon = 0.1$.

- **AdaptiveAgent(Ada)**
  Ada determines the target utility value depending on both time and the opponent's behavior. Setting the maximum utility value at that time of the bid proposed by the opponent $u_{max}^{\text{from } i}$ as the lower bound of the concession, the target utility value is conceded over time. The target utility value for the rth round of the Ada u_r^{target} is determined using Equation (8).

$$u_r^{target} = u_{max}^{\text{from } i} + (1 - u_{max}^{\text{from } i})\left(1 - \frac{r}{R-1}\right)^e \tag{8}$$

5.2 Learning Bidding Strategy

We have A_0 learn the bidding strategy using the proposed method described in Section 4. As a baseline, we have A_0 learn without a representation function.

Hyperparameter Settings

In this experiment, we use actor–critic type PPO (Schulman et al., 2017) as an RL algorithm. To optimize actor and critic parameters, we use Adam (Kingma and Ba, 2014) as the optimization function, with learning rates of 3×10^{-6} for the actor and 1×10^{-5} for the critic. The range ε of the clip of PPO is 0.2. The number of learning epochs for actor and critic updates is 200.

To optimize the parameters of the representation function, we also use Adam as the optimization function, and the number of learning epochs is 2000. λ in Equation (4) is 0.1, the length of the embedding vector output by the representation function is 16, and the learning rate of the representation function is 10^{-6}.

Learning Flow

The agent learns by repeating the following simulations. One simulation runs for 50 days with a negotiation round limit of 20. At the beginning of each day, the interaction episodes of the previous day are input into the representation function, and the average of the outputs is used as the strategy model of the negotiation partner. The state, action, and reward per round are added to the buffer for each opponent. At the end of the day, the interaction episodes are added to the episode buffer for each opponent. The policy π_ϕ is updated by PPO using data stored in buffers at the end of the simulation, and the representation function f_θ is updated through unsupervised learning using data stored in episode buffers for every 10 simulations.

The learning agent prioritizes bids aligning with a target unit price, irrespective of quantity. It always proposes a bid whose quantity is 1 and continues to negotiate despite exceeding raw material quantities from exogenous contracts.

5.3 Evaluation of RL Agent

Experimental Settings

For A_1, A_2, A_3, A_4, we perform 100 times simulations in all 35 environments by selecting four agents from TiD, CTD, TFT, and Ada, allowing for duplication. As baselines for evaluating the performance of the agent learned using the proposed framework with the representation function, we use the agent learned by the same RL method without the representation function and the four types of agents with heuristic strategies, TiD, CTD, TFT, and Ada. We use the two types of the number of simulations during learning, $N = 3000$ and 30000, for both agents learned using the proposed framework and without the representation function.

In the simulation for the evaluation, an RL agent proposes a bid whose unit price corresponds to $u_{target}^{\text{for } i}$ according to Equation (2) and whose quantity is the smaller of its required quantity based on the exogenous contract and its agreed contracts or the quantity proposed by the opponent agent just before. It only accepts a bid in which its utility value calculated by Equation (2) is not lower than $u_{target}^{\text{for } i}$ of its next proposal and its quantity is not more than the quantity it needs based on the exogenous contract and its agreed contract. In addition, they terminate negotiation when the total quantity of intermediate products contracted through negotiations exceeds the quantity of raw materials received under the exogenous contract.

Evaluation Indicators

We use four indicators to evaluate the experimental results.

- **Final Score:** The profit rate.
- **Agreement Rate:** The rate of days an agreement is formed with each type of opponent agent.
- **Agreement Utility:** The averaging utility value of the agreed bids with each type of opponent agent.
- **Intra-Inter Clustering Ratios(IICR):**
  IICR is the average distance between points belonging to the same cluster relative to the average distance between points belonging to different clusters. If there are n clusters of points in the space, and T_i is the set of n_i points in cluster i, the IICR is calculated using Equation (9).

$$\text{IICR} = \frac{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{n_i^2}\sum_{a=1}^{n_i}\sum_{b=1}^{n_i}||t_a^{(i)} - t_b^{(i)}||_2}{\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n}\frac{1}{n_i n_j}\sum_{a=1}^{n_i}\sum_{b=1}^{n_j}||t_a^{(i)} - t_b^{(j)}||_2} \quad (9)$$

  In this study, we use the IICR to evaluate the robustness of embedding vectors output by the representation function.

5.4 Experimental Results & Discussion

Evaluation by Final Score

We show the mean values of the final scores of the agent learned using the proposed framework with the representation function (Emb), the agent learned

without the representation function (No Emb), TiD, CTD, TFT, and Ada in Table 1. As listed in Table 1, regardless of whether the representation function is used, RL agents earned higher scores than heuristic agents (TiD, CTD, TFT, Ada). Therefore, by the formulation proposed in this study, RL agents can learn effective strategies, regardless of whether the representation function is used.

As shown in Table 1, the agent that learned 30000 simulations without the representation function earned a higher score than the agent that learned 30000 simulations with the representation function. We discuss the reasons for this. Because the representation function is learned with discriminative representation as one of its objectives, it outputs a discriminative vector for each opponent agent. This vector enables the agent to perform RL by distinguishing between the strategies of its opponents. In this case, because the input to the representation function is the state of the negotiation with each agent, the information obtained from the output of the representation function can be regarded as latent information in the state. Therefore, we consider that the RL of 30000 simulations without the representation function achieved a high final score in the experiment because the agents learned while distinguishing the strategies of opponents directly from the state. Conversely, RL by adding the embeddings output by the representation function to the state would have increased the size of the state space. Thus we could not obtain a very good strategy with representation functions.

Evaluation by Agreement Rate

We show the mean values of the agreement rates of RL agents and heuristic agents for each opponent agent in Table 2.

As listed in Table 2, the agent learned without the representation function had a lower agreement rate with CTD than the agent learned with the representation function, TiD, CTD, and Ada. As shown in Table 2, the agent learned with the representation function had a higher agreement rate with CTD than the other agents whether the number of simulations. One of the reasons that the agent learned without the representation function had a lower agreement rate with CTD is that it had obtained a too-bullish strategy. Because it learns to earn a high utility in negotiations with any agents, including TiD, with which the maximum utility it can earn upon agreement is higher than that with CTD, the agent learned without the representation function frequently proposes a bid whose utility value for it is higher than that with CTD can accept. Conversely, the agent learned with the representation function had a high agreement rate with CTD because the embedding vector output by the representation function enabled it to learn to distinguish CTD from the other agents, including TiD.

As shown in Table 2, the agent learned for 30000 simulations with the representation function had a lower agreement rate with TFT than the agent learned for 30000 simulations without the representation function. One of the reasons for this is that the states taken in negotiations with TFT are significantly different from the states taken in negotiations with the other agents. If the RL agent can distinguish TFT from the other agents only by the state, the embedding vector output by the representation function is redundant and prevents learning.

Evaluation by Agreement Utility

We show the mean values of the agreement utilities of RL agents and heuristic agents for each opponent agent in Table 3.

As listed in Table 3, regardless of the presence or absence of the representation function, an agent learned for 30000 simulations earned a higher agreement utility than an agent learned for 3000 simulations. Therefore, in the early stage of learning, an agent learns to increase the agreement rate and then learns to increase the agreement utility.

As shown in Table 3, the agent learned with the representation function earns higher agreement utility in the negotiation with CTD and Ada, whereas the agent learned without representation function earns higher agreement utility in the negotiation with TiD and TFT. In negotiations with CTD and Ada, the state transitions are similar to those of TiD. However, CTD differs from TiD in the acceptable utility value in the end, and Ada differs from TiD in that it becomes more aggressive when its opponent agent concedes. Therefore, by introducing the representation function and learning in a more discriminative manner, it obtained higher agreement utility. Conversely, TiD is a simple strategy, making it relatively easy to learn effective strategies. The state transitions in negotiations with TFT are significantly different from those taken in negotiations with other agents. Therefore, in negotiations in TiD and TFT, the agent learned without the representation function obtained higher agreement utility because the embedding vector was redundant.

Evaluation of Robustness of Embeddings

We show the mean values of IICR of the embedding vector output by the representation function trained in 30000 simulations in the case more than one type of agent is included in the opponents in Table 4. IICR is smaller if the opponents include TFT and larger

Table 1: Means of the final scores.

Emb ($N = 3000$)	Emb ($N = 30000$)	No Emb ($N = 3000$)	No Emb ($N = 30000$)	TiD	CTD	TFT	Ada
1.16	1.24	1.19	1.47	1.03	1.04	0.97	1.04

Table 2: Means of agreement rates with each opponent.

A_0 \ Opponent	TiD	CTD	TFT	Ada
Emb($N = 3000$)	0.93	0.92	0.91	0.85
Emb($N = 30000$)	0.93	0.90	0.85	0.92
No Emb($N = 3000$)	0.92	0.83	0.92	0.82
No Emb($N = 30000$)	0.93	0.84	0.93	0.80
TiD	0.89	0.89	0.88	0.89
CTD	0.89	0.89	0.89	0.89
TFT	0.89	0.81	0.0	0.17
Ada	0.89	0.89	0.88	0.89

Table 3: Means of agreement utilities with each opponent.

A_0 \ Opponent	TiD	CTD	TFT	Ada
Emb ($N = 3000$)	0.66	0.72	0.64	0.72
Emb ($N = 30000$)	0.73	0.77	0.65	0.77
No Emb ($N = 3000$)	0.85	0.66	0.72	0.27
No Emb($N = 30000$)	0.97	0.70	0.84	0.59
TiD	0.47	0.43	0.68	0.36
CTD	0.52	0.48	0.68	0.44
TFT	0.33	0.36	0.00	0.23
Ada	0.57	0.51	0.60	0.46

if they include both CTD and Ada. Therefore, although the representation function discriminates TFT more strongly, it cannot accurately discriminate between CTD and Ada.

6 CONCLUSION & FUTURE WORK

In this study, we proposed an RL framework that introduces representation learning for concurrent negotiation and formulated a concurrent negotiation problem. The proposed RL framework considers the differences in opponent strategies using a representation function that is learned unsupervised from negotiation log data with each opponent agent for generative–discriminative representation. We defined states, actions, and rewards using utility values and conducted evaluation experiments using agents learned via RL using the proposed framework. The results showed that the RL agents scored higher values than the heuristic agents. Although the agent learned with the representation function earned a lower final score than the agent learned without the representation function, in certain cases, there was an improvement in the agreement rate and agreement utility with each agent.

Table 4: IICR(Best 3 and Worst 3).

Opponents	IICR	Opponents	IICR
TFT-Ada-Ada-Ada	0.26	CTD-CTD-Ada-Ada	0.94
CTD-CTD-TFT-TFT	0.26	CTD-CTD-CTD-Ada	0.93
TFT-TFT-TFT-Ada	0.26	CTD-Ada-Ada-Ada	0.93

One of our future works is to investigate a method for advantaged negotiation even in a test environment where opponent agents use different strategies from one used in learning. In general, the opponents' behaviors are unknown in advance, so it is necessary to establish a method for efficiently fine-tuning in the actual test environment after learning it in advance using opponents with various strategies.

REFERENCES

Baarslag, T., Aydoğan, R., Hindriks, K. V., Fujita, K., Ito, T., and Jonker, C. M. (2015). The automated negotiating agents competition, 2010–2015. *AI Magazine*, 36(4):115–118.

Bakker, J., Hammond, A., Bloembergen, D., and Baarslag, T. (2019). Rlboa: A modular reinforcement learning framework for autonomous negotiating agents. In *The 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 260–268.

Grover, A., Al-Shedivat, M., Gupta, J., Burda, Y., and Edwards, H. (2018). Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Mohammad, Y., Greenwald, A., Fujita, K., Klein, M., Morinaga, S., and Nakadai, S. (2022). Supply chain management league (oneshot). http://www.yasserm.com/scml/scml2022oneshot.pdf.

Mohammad, Y., Nakadai, S., and Greenwald, A. (2021). Negmas: A platform for automated negotiations. In *PRIMA 2020: Principles and Practice of Multi-Agent Systems: 23rd International Conference, Nagoya, Japan, November 18–20, 2020, Proceedings 23*, pages 343–351. Springer.

Mohammad, Y., Viqueira, E. A., Ayerza, N. A., Greenwald, A., Nakadai, S., and Morinaga, S. (2019). Supply chain management world: a benchmark environment

for situated negotiations. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems: 22nd International Conference, Turin, Italy, October 28–31, 2019, Proceedings 22*, pages 153–169. Springer.

Rubinstein, A. (1982). Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society*, pages 97–109.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.