Deep reinforcement learning for acceptance strategy in bilateral negotiations

Razeghi, Yousef; Yavuz, Ozan; Aydogan, Reyhan

DOI
10.3906/ELK-1907-215

Publication date
2020

Document Version
Final published version

Published in
Turkish Journal of Electrical Engineering and Computer Sciences

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1824 – 1840

doi:10.3906/elk-1907-215

Deep reinforcement learning for acceptance strategy in bilateral negotiations

Yousef RAZEGHI[1], **Ozan YAVUZ**[1], **Reyhan AYDOĞAN**[1,2*]
[1]Department of Computer Science, Özyeğin University, İstanbul, Turkey
[2]Interactive Intelligence Group, Delft University of Technology, Delft, Netherlands

Received: 28.07.2019 • **Accepted/Published Online:** 04.03.2020 • **Final Version:** 29.07.2020

Abstract: This paper introduces an acceptance strategy based on reinforcement learning for automated bilateral negotiation, where negotiating agents bargain on multiple issues in a variety of negotiation scenarios. Several acceptance strategies based on predefined rules have been introduced in the automated negotiation literature. Those rules mostly rely on some heuristics, which take time and/or utility into account. For some negotiation settings, an acceptance strategy solely based on a negotiation deadline might perform well; however, it might fail in another setting. Instead of following predefined acceptance rules, this paper presents an acceptance strategy that aims to learn whether to accept its opponent's offer or make a counter offer by reinforcement signals received after performing an action. In an experimental setup, it is shown that the performance of the proposed approach improves over time.

Key words: Deep reinforcement learning, automated bilateral negotiation, acceptance strategy

1. Introduction

Automated negotiation [1] is an important study field in artificial intelligence, where intelligent agents negotiate on behalf of their users on multiple issues with the aim of maximizing their own utility. Bidding strategy [2–4], opponent modeling [5–7], and acceptance strategy are the main challenges in automated negotiation. Agents exchange offers consecutively between each other to reach an agreement in a given negotiation scenario. This interaction is governed by a certain protocol determining the rules of encounter. The alternative offers protocol [8] is 1 of the most widely used protocols in bilateral negotiation. According to this protocol, an agent initiates the negotiation with an offer and its opponent can accept or reject this offer. If the opponent accepts the current offer, negotiation ends with an agreement and the utility of the agreement for each agent is calculated with respect to their preference profiles. Otherwise, the opponent agent takes the turn and makes a counter offer. This process continues in a turn-taking fashion until an agreement is reached or the negotiation deadline for the session expires. If there is no agreement at the end of the negotiation, each agent gets the reservation value (i.e. the best alternative to a negotiated agreement, abbreviated as BATNA). That is the utility of the best alternative for the negotiating party if the negotiation fails to result in an agreement. The turn-taking fashion of taking actions in automated negotiation makes it an appropriate environment for applying reinforcement learning [9] (RL), where the agents can learn the best actions to be taken based on the feedback given during these interactions.

RL is the process of finding an optimal policy in an environment based on feedback received from the environment in response to the agents' actions. In other words, agents learn from their experiences. In RL,

*Correspondence: reyhan.aydogan@ozyegin.edu.tr

there are a number of states that the agent may be situated in. An action is any possible move that the agent can make in the given state. The goal of the agent is to select the action maximizing its expected value. An RL agent receives feedback from its environment for each action it takes and this feedback can be formulated as a reward. The agent has to learn the outcomes of its actions in the long run, which is known as the credit assignment problem. In RL problems, an environment is modeled as a Markov decision process (MDP) with inputs (actions taken by the agent) and outputs (observations and rewards sent to the agent). The agent may aim to learn a policy that aims to maximize its reward while interacting with the environment. In this work, our negotiating agent employs RL in order to determine whether or not it should accept its opponent's current offer. Accordingly, it accepts the given offer or makes a counter offer.

The existing works on acceptance strategies for automated negotiation are mostly based on predefined rules, which take remaining time and/or utility of the negotiation outcome into account. For instance, AC-next [10] is 1 of the most widely used acceptance strategies, where an agent accepts its opponent's offer if the utility of the opponent's offer is higher than or equal to the utility of its own next offer. Furthermore, these predefined rules can be combined to form more complex acceptance strategies as proposed by Baarslag et al. [10]. In the present work, we aim to develop an acceptance strategy that learns when to accept the opponent's offer using RL, while some recent works [11, 12] employ RL in order to learn what to bid. The sequential flow of operations in the negotiation framework makes agent-based negotiation an appropriate field to implement and assess the RL algorithms on agents. Since our aim is to design and develop a domain-independent acceptance strategy applicable to any given negotiation scenarios, GENIUS [13] (**G**eneric **E**nvironment for **N**egotiation with **I**ntelligent multi-purpose **U**sage **S**imulation) environment is chosen as our negotiation platform. An advantage of the GENIUS environment involves the BOA (bidding, opponent model, accepting) framework [14], which enables developing negotiation components (i.e. bidding strategy, opponent modeling, and acceptance strategy) separately. This framework enables researchers to develop and study individual parts of the negotiation strategy.

The primary goal of our approach is the development of an acceptance strategy that can negotiate and achieve reasonable results with different opponents regardless of the negotiation domain. During the negotiation, it is observed that our agent learns what to bid and when to accept its opponents counter offer and improves its performance over time. The main contribution of this paper is designing and developing a domain-independent acceptance strategy using RL. The developed strategy is compared with AC-next acceptance strategy and in some cases the proposed approach performs better than AC-next, while in other cases the performance is similar to AC-next.

The rest of this paper is organized as follows: Section 2 explains the proposed acceptance strategy by providing the definitions and formulation of problem used for adapting RL. Section 3 describes the experimental setup, demonstrates the achieved results in both training and testing negotiation sessions, and also includes an explanation of the methodologies used for generalization and regularization of the model. Section 4 briefly discusses the related works. Section 5 concludes the paper and discusses future work.

2. Proposed acceptance strategy

Our negotiation environment consists of 2 agents negotiating on multiple issues (e.g., travel destination, location, and date) within a certain time limit. The agents take their actions in a turn-taking fashion by following the *alternating offers protocol* [8]. There are 2 possible actions, which consist of accepting or rejecting the opponent's offer. The preferences of agents are represented by means of an additive utility function [15], where overall utility of an outcome is calculated by the weighted sum of the each individual utility value of each issue.

In our negotiation setup, the negotiating agents can only access their own utility function. That is, they do not know their opponent's utility function. If the agents reach an agreement, they receive the utility of that agreement before the deadline. Otherwise, each agent gets the reservation value as the final utility. The goal of the agents is to maximize their utility gained at the end of the negotiation.

In the present work, we use a variant of the Q-learning algorithm, called deep Q-network (DQN), to learn what to bid and when to accept. We first explain the fundamentals of Q-learning and then present how we adopt the DQN in negotiation. Q-learning is an off-policy reinforcement learning algorithm in which an agent receives a reinforcement signal called immediate reward after performing an action and aims to maximize its total reward. In Q-learning, agents try to learn the values (Q values) of each state-action pair. That means the agent estimates the value of taking an action a in a given state s. For this purpose, it maintains a table of state-action values. The agent learns the values of its actions with Q-learning updates and performs its actions based on ϵ-greedy policy. ϵ is a constant value between 0 and 1 that determines the probability of choosing a random action and 1 - ϵ determines the probability of choosing an action where its action value is the maximum. By this method, for each action that the agent performs, it explores the environment with ϵ probability and exploits the environment based on its experience with 1 ϵ probability.

When our agent makes an offer, there is uncertainty about the acceptance of the offer by the opponent. Because the opponent's response to our agent's offer is influential in the step transitions, our agent waits for the opponent's response to update the Q function after making an offer. After each state transition, the Q function is updated as follows:

$$Q(S,A) \leftarrow Q(S,A) + \alpha[r + \gamma max_a Q(S^{'},A) - Q(S,A)]. \tag{1}$$

In Equation 1, the $Q(S,A)$ is the Q function yielding the overall expected reward of doing action A in state S. S' is the transitioned state and r stands for the immediate reward received by transitioning from state S to S' after taking the action A. γ represents a constant, namely the discount factor, which scales the temporal difference (TD) approximation. The expression $[r+\gamma max_a Q(S',A)-Q(S,A)]$ is called the TD error, which is the magnitude of Q value update on $Q(S,A)$ and is scaled by the α, which is the learning rate.

Keeping track of state-action values by using the tabular approach is infeasible when the state space is vast. To remedy this, we propose to use DQN in which a neural network is used to approximate the Q function with artificial neural networks (ANNs) for learning when to accept in negotiation. The DQN takes the state as the input and values of actions as the output.

For the proposed acceptance strategy in bilateral negotiations, a state can be represented as a 5-tuple as follows: $< \Delta O, D, MNU, R, C >$, where

- ΔO: The difference between the reservation value and the utility of the received offer made by the opponent. It is a real number between between –1.0 and 1.0.
- D: Scaled remaining time. It is a real number between 0 and 1. The value of 1 denotes the end of the negotiation (i.e. negotiation deadline).
- MNU: The utility of the agent's next offer, a real number between zero and one.
- R: The target utility, which is the minimum utility that the agent aims to gain. R is taken as 0.8 in our work.
- C: Current utility value of the opponent's offer. It is a real number between 0 and 1. Higher values mean that the offer is more preferred by the agent.

Note that the target utility is the minimum utility value that the agent aims to achieve at the end of the negotiation to get a positive return. The received utility, on the other hand, is the utility value the agent gets at the end of the negotiation. The immediate reward that the agent receives after each step transition is calculated as follows:

$$r = \begin{cases} -2^{|t-f|} & t > ru \\ 2^{|t-f|} & t < ru \\ 0 & \textit{the step transition is nonterminal}, \end{cases} \tag{2}$$

where

- t : Target utility
- f : Final utility
- ru : Received utility

The state transitions result in *terminal* or *nonterminal* states. Nonterminal states are those in which the agents have not reached an agreement yet and continue negotiating. Therefore, the reward is zero for nonterminal states. Terminal states are the states in which 1 of the agents accepts an offer from its opponent or when the negotiation deadline expires without an agreement. The reward in the terminal states is calculated as shown in Equation 2.

As illustrated in Figure 1, there is a feedback loop between the agent and the environment. The current state S is the input to the neural network and the output is 2 numbers indicating the expected reward of performing actions, namely accept and reject. A state transition happens when an action is performed. Then the next action is performed with respect to the new state with the same process. This feedback loop shapes the learning process of the model.

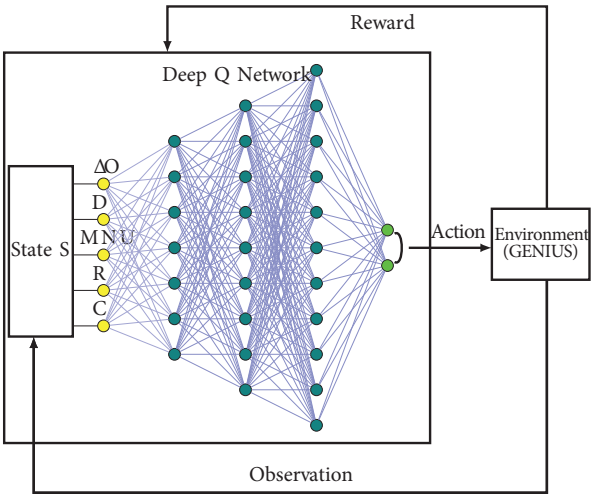


Figure 1. Proposed negotiation architecture.

Algorithm 1 illustrates the proposed approach for our acceptance strategy where NN stands for neural network.

Determine_Acceptability() is called after receiving an offer from the opponent and our agent makes its decision based on the RL-based acceptance strategy. The DQN is trained after an action is performed. Note that at the initial state (e.g., when no party has made any counter offer), the previous state is null and the action is performed without any training (lines 3–5). After the initial state, first, the DQN is trained with respect to the Q-learning update rule as shown in Equation 1 and then the action is performed. Note that the DQN is trained in the code block as shown between lines 2 and 15, and the action is performed as shown between lines 16 and 27. The agent explores with a probability of ϵ, which means taking a random action (lines 24–27). Furthermore, it exploits and chooses the best action according to the Q-function with a probability of $1-\epsilon$ (lines 16–23).

Algorithm 1 Acceptance strategy based on Q-learning.

```
1:  procedure DETERMINE_ACCEPTABILITY()
2:      < V_A, V_R > ← NN.predict(S_Cur)
3:      if S_Prev == null then
4:          S_Prev ← getCurrentState()
5:      else
6:          < V_APrev, V_RPrev > ← NN.predict(S_Prev)
7:          if V_A > V_R then
8:              Act_Max ← Accept
9:          else
10:             Act_Max ← Reject
11:         end if
12:         r ← 0
13:         V_RPrev ← r + γ * S_Cur[Act_Max]
14:         NN.train(S_Prev, < V_APrev, V_RPrev >)
15:     end if
16:     With 1-ϵ probability do:
17:     Begin
18:         if V_A > V_R then
19:             return Accept
20:         else
21:             return Reject
22:         end if
23:     End
24:     With ϵ probability do:
25:     Begin
26:         return Randomly(Accept|Reject)
27:     End
28: end procedure
```

- V_A: The Q-value of accept action in the current state
- V_R: The Q-value of reject action in the current state
- NN: The neural network
- S_{Cur}: Current state
- S_{Prev}: Previous state

- A_{Prev}: The Q-value of accept action in the previous state
- R_{Prev}: The Q-value of reject action in the previous state
- Act_{Max}: The action with the maximum Q-value

Since our aim is to propose an acceptance strategy based on deep reinforcement learning that can generalize over several domains and opponent agents during bilateral negotiations, we first trained the model of the agent while negotiating with several agents in different negotiation scenarios (i.e. domain with 2 conflicting preference profiles). The problem with training by letting our agent negotiate with different opponents in a variety of domains is that the agent is specialized in terms of the current opponent in the given domain (i.e. overfitting the current setup). That is, the agent forgets his previous negotiation experiences with other agents in other domains.

Due to environmental constraints, setting sessions that can include different domains and several opponents at the same time is impossible. We overcome this problem by generating virtual states and feeding them through experience replay memory, which is used already for previous negotiation sessions. Experience replay memory is a stack in which state transitions with rewards belonging to the previous and current episodes are stored. In this approach, the DQN is fed by taking random samples from this memory. To prevent overfitting in the DQN, we applied L2 regularization to the DQN, which adds the squared magnitude of neural network weight coefficients as a penalty term to the loss function of the DQN.

3. Experimental evaluation

In order to evaluate the performance of the proposed approach, we implemented a negotiating agent adopting our RL-based acceptance strategy in the BOA framework of the GENIUS environment. This environment hosts a variety of negotiation scenarios (i.e. negotiation domain and a pair of preference profiles) and negotiating agents. In this platform, an agent can negotiate with an opponent in different negotiation scenarios. Recall that we adopt experience replay memory to maintain the state and action pairs from different negotiation sessions. Therefore, our agent using the DQN learns the optimal acceptance conditions in different negotiation settings. In the following sections, we will describe how we trained our RL-based agent (Section 3.1) and present the experiment results with respect to the performance of the agent in the test environment (Section 3.2).

3.1. Training session

For training purposes, a well-known negotiation domain *England–Zimbabwe* [16] is used. In this scenario, there are 576 possible outcomes in total. Figure 2 demonstrates the utility distribution of available bids for this scenario, where the Nash point is (0.91, 0.73). The difficulty level to reach an agreement is considered medium. For each profile, our agent negotiates with its opponent 600 times. That is, the training data involve 600 negotiation sessions (i.e. epochs). Note that our agent plays both sides (600 times for England and 600 times for Zimbabwe); it makes 1200 negotiation sessions in total. The deadline for the negotiations is 180 s. If agents cannot reach an agreement before the deadline, they receive a zero utility.

In our setup, the opponent agent employs the *Gahboninho* [17] negotiation strategy. This agent is selfish and stubborn. At the beginning for a certain period, this agent insists on making bids with the utility of 0.9. Afterwards, the agent becomes more selfish and hardheaded and at the very end of the negotiation Gahboninho concedes to avoid disagreement. Recall that an agent consists of the following components in the BOA framework:

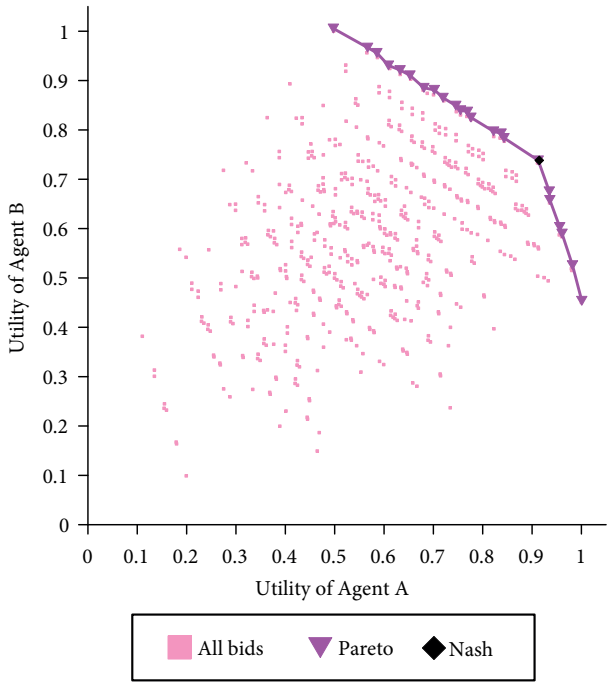


Figure 2. Utility distribution for England–Zimbabwe.

- **Bidding strategy**: This model defines the amount of concessions with respect to negotiation flow; thus, it determines the estimated utility of its next offer and what bid will be offered.
- **Opponent modeling**: A learning model is designed to predict the preference profile of the opponent and/or the opponent's strategy based on the exchanged bids during the negotiation.
- **Acceptance strategy**: It determines whether to accept the received bid from the opponent.

As a bidding strategy, we pick the bidding strategy of AgentK [18]. AgentLG and NTFT (i.e. not tit-for-tat) are used for the opponent preference modeling and opponent strategy modeling, respectively. As an acceptance strategy, our agent implements the RL-based acceptance strategy proposed in the present paper.

Our agent starts the negotiation by taking random actions and explores the action space during the negotiation session in order to find the optimal acceptance strategy over time. The agent decreases the exploration ratio over time and it gradually exploits its learned model and tries to take the action giving maximum expected rewards.

In the training phase, we first analyze the utility of the agreements. Figure 3 demonstrates the average utilities of the agreements over 10 negotiation sessions for our agent negotiating with *Gahboninho* for both preference profiles. The graph on the left-hand side shows the results when our agent negotiates under preference profile-1. Similarly, the graph on the right-hand side shows the results while negotiating under preference profile-2. The dashed red horizontal line in Figure 3 shows the average of all utilities for the 600 negotiations. In the rest of the paper, we use the same notation while reporting our results. Note that each session point in the figure represents ten consecutive negotiation sessions. Recall that our agent negotiates with the same opponent 600 times for each preference profile. It can be seen that the performance of our agent improves after getting a certain amount of experience (i.e. 25 negotiation session points in this case). The increasing trend in terms

of received utility indicates the learning capability of our agent. Furthermore, it can be noted that there are some fluctuations in the utilities of agreements, especially after the significant rise in utility. Recall that most of the bidding strategies have a stochastic nature. Therefore, even the same agents negotiating with the same opponent may end up with a different negotiation outcome. This fluctuation can be explained due to the stochastic nature of bidding strategies. Another observation is that our agent learned to reject offers at the early phases of the negotiations to get better offers from the opponent over time. Note that our opponent starts conceding when approaching the deadline.

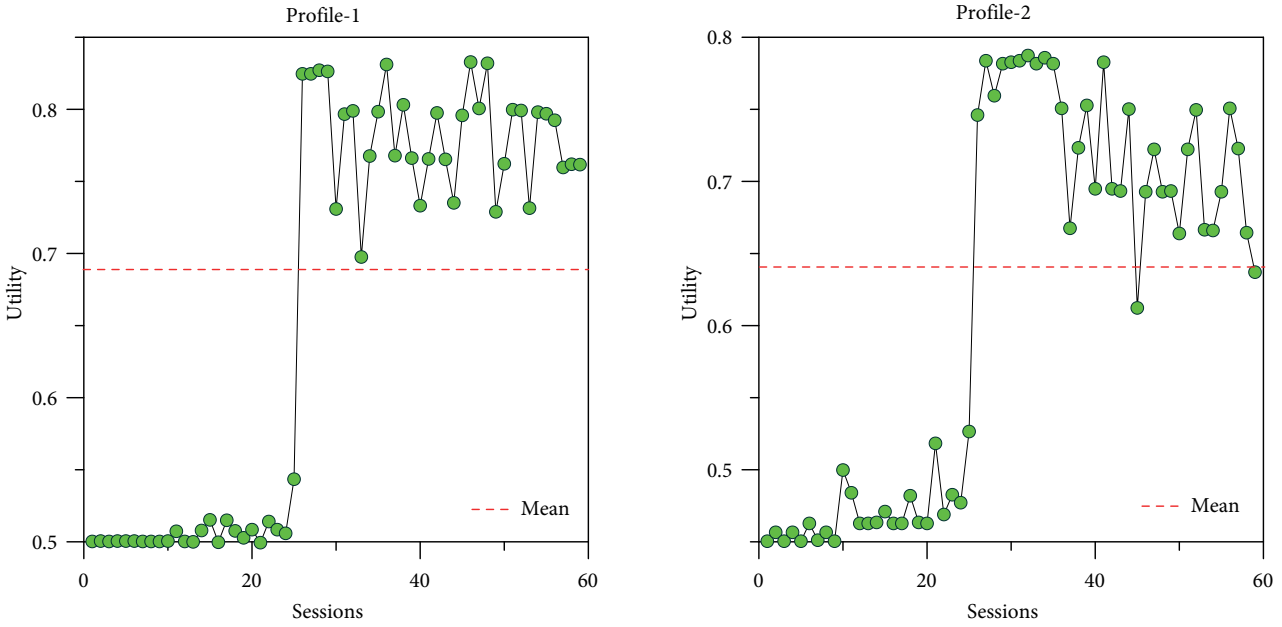


Figure 3. Utility changes in the *RL acceptance party*.

As a baseline negotiation strategy, we used an agent that randomly negotiates. Figure 4 shows the average utility changes of the agreements for the randomly negotiating agent. When we compare these results with those obtained with the RL acceptance strategy shown in Figure 3, it is obvious that our agent receives much higher utility.

Furthermore, we also analyze the negotiation outcome in terms of distance to the Nash product solution (i.e. the bid whose utilities for both parties is the maximum), in terms of distance to Pareto solutions (i.e. the bid whose utility cannot be improved for 1 party without worsening the utility of the other party), and social welfare (i.e. sum of agents' utilities). Assessment of the training session results reveals a noticeable change in the agent's behavior in terms of Nash, Pareto, and social welfare metrics. Figure 5 demonstrate the utility changes for our agent according to different performance metrics. As in Figures 5a and 5b, it is observed that the distance to Pareto is close to zero in the former negotiation sessions, while the distance increases over time. Note that having a zero distance to Pareto means that the agreement is 1 of the Pareto optimal outcomes. It is worth noting that if the utility of our opponent is the highest (no matter what we received), it is automatically a Pareto optimal outcome. Recall that an outcome is Pareto optimal if the agents cannot improve the utility of the outcome for 1 of the agents without worsening the others. Since in the former sessions our opponents aim to get the best offer for themselves, the distance to Pareto is around zero. Recall that the best offers for each agent are always Pareto optimal outcomes according to the definition of Pareto optimal solutions. However, it increases when our agent starts learning when to accept.

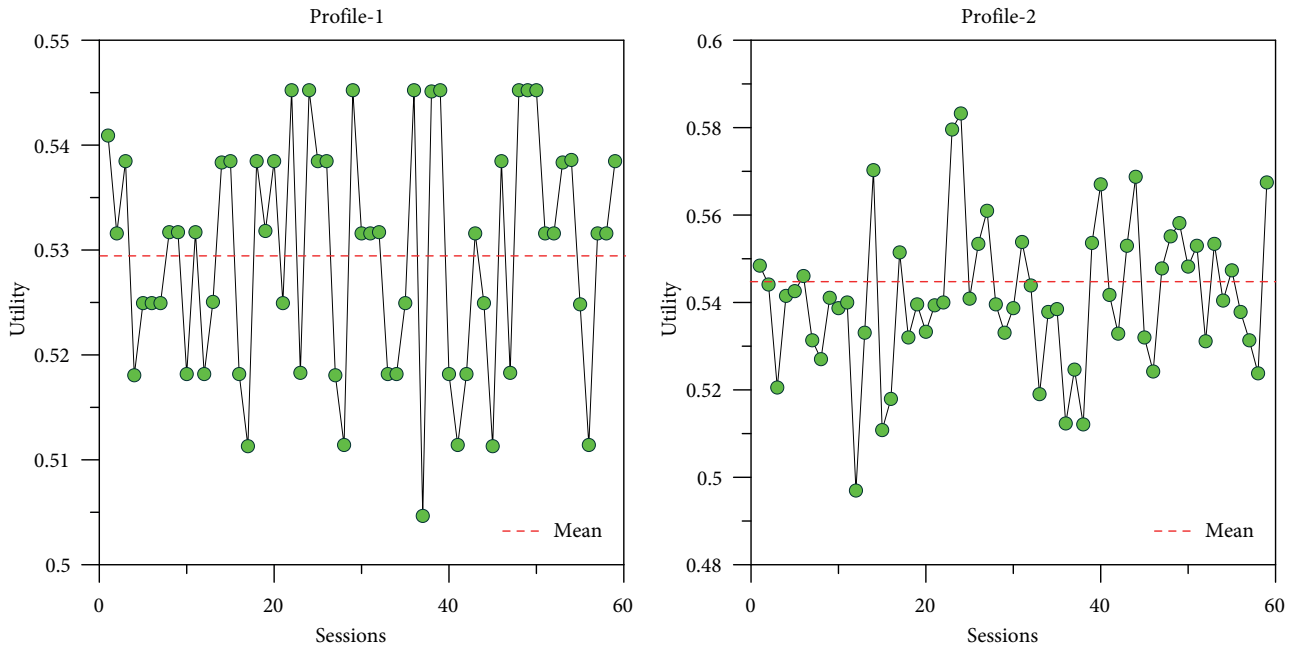


Figure 4. Utility changes in the *random behavior agent*.

Recall that the Nash product bid in the negotiation is the bid where the product of the utilities for each agent is the maximum. The distance to the Nash solution (Euclidean distance) is almost steady in the initial negotiations. Because our agent explores more than exploits its experience, it accepts its opponent's offer too early and the opponent gets the utilities close to 1. However, as our agent's acceptance strategy improves, fluctuations occur because the variance in the final utilities increases, as shown in Figures 5b-5e. Similar applies to the social welfare too as shown in Figures 5c-5f. Note that the social welfare is the sum of the agents' utilities.

3.2. Test session

In automated negotiation, it is important to design agents that can negotiate with different opponents. In order to assess the performance of the proposed RL-based acceptance strategy, we tested our agent with various opponents and negotiation scenarios. For opponent strategies, we picked the following 6 strategies: *Agent Smith* [19], *Nozomi* [20], *Yushu* [21], *FSEGA* [22], *IAMHaggler* [23], and *Pars Agent* [24].

These 6 agents were the top rated ones in previous years at the International Automated Negotiating Agents Competition (ANAC) [25]. A brief description of the opponent agent's strategies is provided below:

- **Agent Smith:** This agent models the opponent's preferences during the negotiation. It initially makes the best offer for itself (i.e. offer with the maximum utility). Afterwards, it compromises over time towards the interest of its opponent.
- **Yushu:** Using a combination consisting of the ten last received bids and an estimation about the remaining round, the agent calculates a target utility and makes its offer with that target utility. Note that Yushu also considers the minimum utility value it may accept while making its offers.
- **FSEGA:** It divides the negotiation into 3 phases. In the first 85% of the negotiations, it aims to model its

opponent by analyzing the exchanged bids. In the second phase (85%–95%), it does not concede. In the last phase (95%–100%), FSEGA employs a concession-based strategy due to the time limit and sends bids accordingly that are just higher than the reservation value. This agent always accepts the best available offer; otherwise, it offers a new bid.

- **IAMHaggler:** This agent constructs an opponent model using Bayesian learning. As a starting point the agent offers a bid with the maximum utility and continuously selects a target utility based on various factors such as opponent model, remaining time, and received bid utility.
- **Pars Agent:** The Pars agent employs a bidding strategy that is a combination of time-dependent, random, and frequency-based strategies to make a bid with high utility that is close to the opponent's offers. This behavior increases the possibility of reaching an agreement sooner. This agent took second place in the individual utility category in ANAC2015 [26].
- **Nozomi:** At the beginning, Nozomi sends an offer with the maximum utility. Based on the opponent's last offer and remaining time, it chooses to compromise or insist.

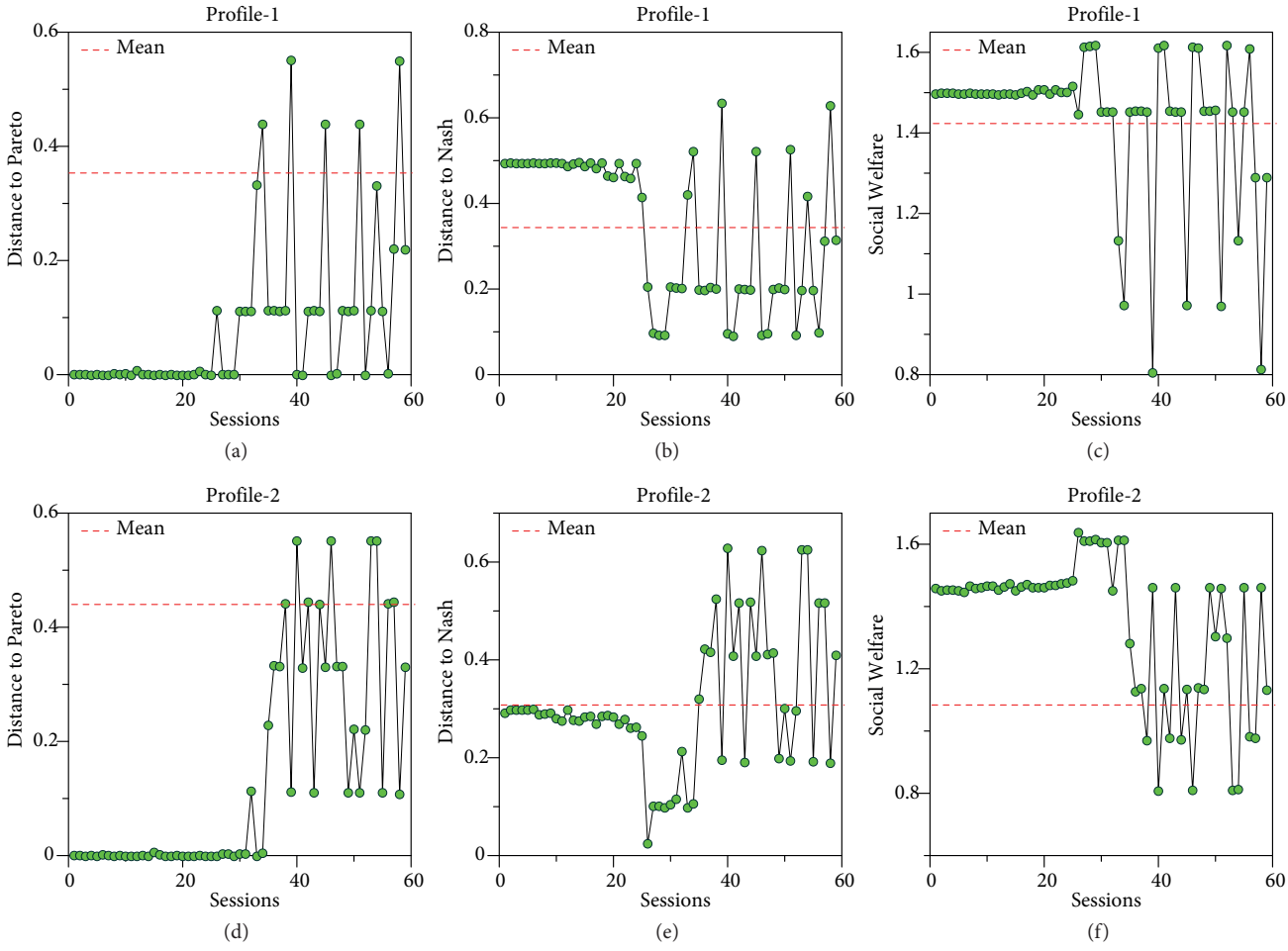


Figure 5. Nash and Pareto metrics of agent behavior.

Three different negotiation scenarios (i.e. party, Amsterdam, and airport) are used to evaluate the performance of the proposed acceptance strategy. We choose domains different from the one used in training in order to evaluate the generalization ability of the proposed RL model. The utility distribution of bids in the test domains is demonstrated in Figures 6a-6b-6c). Note that the Nash product outcomes are denoted by the black marker, while the Pareto efficient outcomes are shown by the pink marker.

In the party domain, there are 6 negotiation issues with varying values (i.e. 4, 4, 4, 4, 3, 4 values for the issues foods, drinks, location, invitations, music, and cleanup, respectively), which result in $4^5 \times 3 = 3072$ possible outcomes, while the Amsterdam trip domain consists of 6 negotiation issues (i.e. 4, 3, 7, 3, 3, 4 values for the issues venue, time of arrival, day of the week, duration, transportation, and souvenirs. respectively) resulting in $3^3 \times 4^2 \times 7 = 3024$ possible outcomes. Although the size of their outcome space is almost the same, utility distributions in those scenarios are different from each other. The airport size selection domain consists of 3 issues (i.e. 10, 7, 6 values for the issues cost, noise, and accident level per million passenger miles, respectively) and $10 \times 7 \times 6 = 420$ possible outcomes exist in this domain. This domain is sparser than the others.

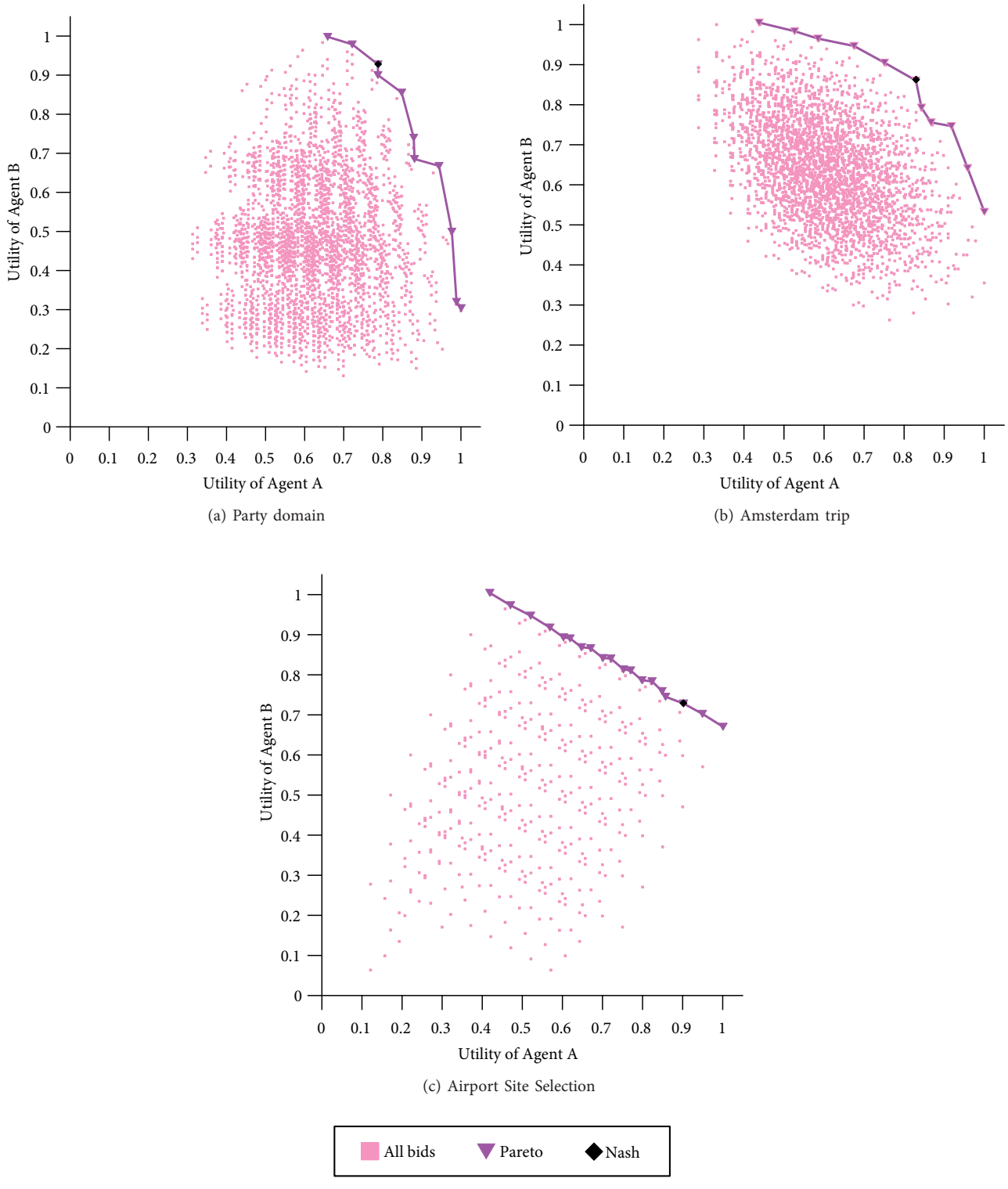
We compare the performance of our acceptance strategy with that of the AC-next [10] acceptance strategy, which is the most widely used acceptance strategy in automated negotiation. When the agents employ the AC-next acceptance strategy, they accept the received offer if its utility is higher than or equal to the utility of the agent's next bid.

In the experiments, we keep the BOA components for bidding strategy, opponent modeling, and opponent modeling strategy the same for those agents. We only change the acceptance strategy to compare their performance. Each negotiation is repeated 10 times and the deadline of each negotiation is set to 10 s. Since each scenario has 2 profiles, each acceptance strategy was tested on both negotiation profiles to assess the overall performance.

Figure 7 shows the average utilities of our agent for both acceptance strategies. According to those results, it can be seen that when our agent negotiates with the opponent except FSEGA, the performance of the RL acceptance strategy is almost the same as that of the AC-next strategy. For instance, while negotiating against Yushu, both agents with AC-next and with our proposed acceptance strategy gained around 0.71. In some cases, the agent with our strategy got higher utility (0.94 versus 0.91 against IAMHaggler), while in other cases the agent with AC-next gained higher utility (0.63 versus 0.72 against FSEGA) on average. The agent reaches agreements with higher utilities when it employs the AC-next strategy for the Amsterdam trip scenario. For the Amsterdam trip domain in Figure 6, it seems that most of the bids are distributed on the right top side of the outcome space. In other words, most of the bids have high utility for both sides. We observed that in many cases agents fail to reach an agreement when our agent employs the RL acceptance strategy for the Amsterdam trip scenario. On the other hand, the RL acceptance strategy outperforms the AC-next strategy for the party and airport site selection scenarios where the utility of the bids is distributed sparsely.

4. Related Work

Bakker et al. recently introduced a reinforcement learning framework for automated negotiating agents [27]. They focus on learning the bidding strategy (i.e. what to bid rather than what to accept). In particular, the agent aims to learn to determine the utility interval of its next offer by considering the utility of its previous and current offers, the utility of its previous and current opponent's offers, and time. They define ten utility intervals, called *bins* (e.g., [0-0.1], [0.1-0.2], ..., [0.9-1.0]) and the agent tries to learn which interval it should



(a) Party domain

(b) Amsterdam trip

(c) Airport Site Selection

Figure 6. Domain information.

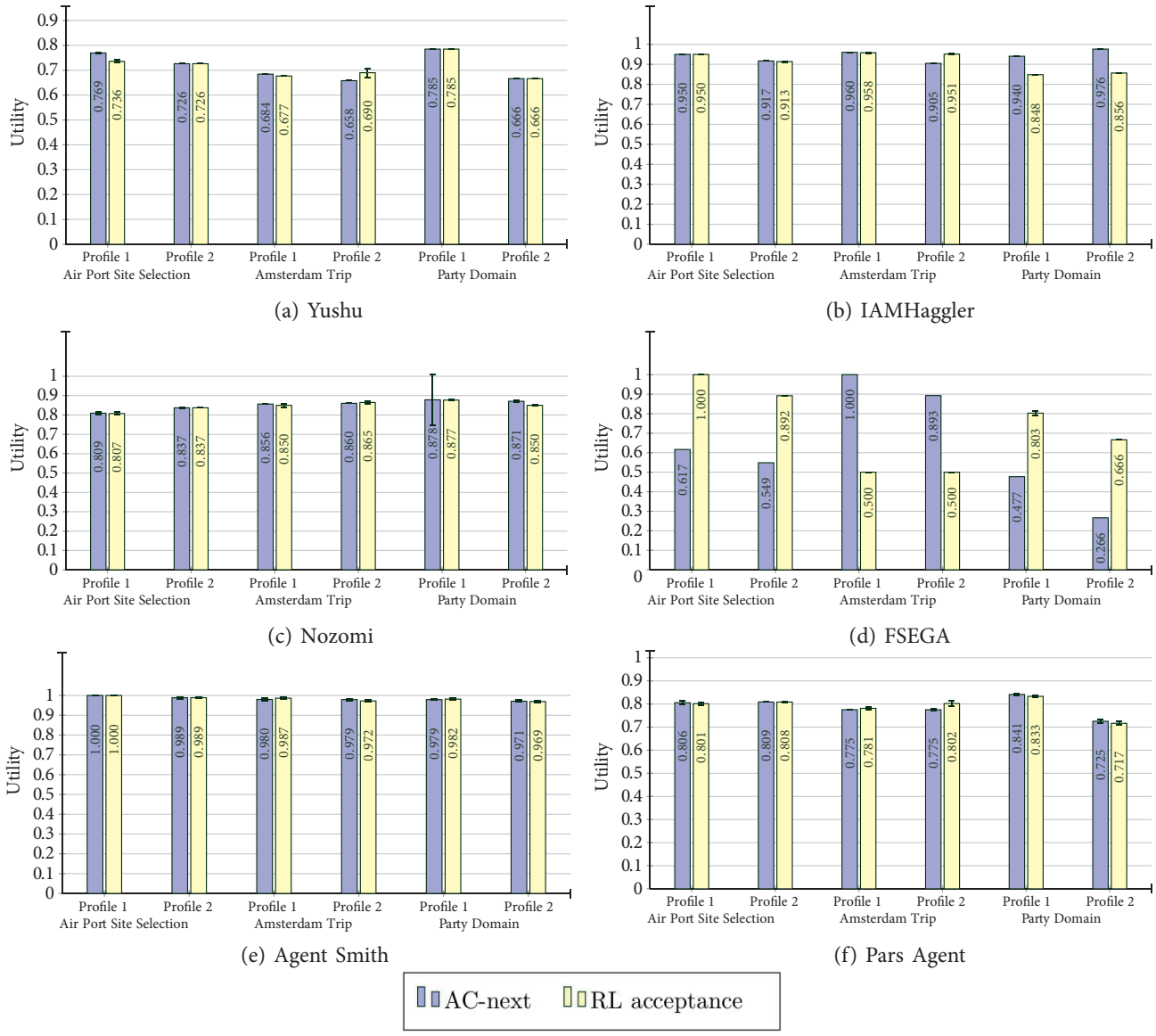


Figure 7. Test results compared to AC-next acceptance strategy.

use for preparing its next offer. While they have a tabular learning approach by using Q-learning, we prefer to adopt a deep reinforcement learning approach by using the DQN. Furthermore, we aim to learn when to accept rather than what bid to make.

Papangelis et al. [28] use RL to learn a multiissue negotiation policy to design an agent negotiating with humans. They use Q-learning with function approximation and consider different feature-based representations of state and action space to handle the large state space. They trained their agent against a simulated user (SU), which was a hand-crafted negotiation agent based on the agenda paradigm [29]. Their reward function is similar to ours; they give a penalty if no agreement is reached before the deadline. If an agreement is reached, the agent receives a reward based on the values of the final offer and the agents preference. They use Q-learning

in order to find an optimal policy. Similar to our approach, they use greedy in the limit with infinite exploration (*GLIE*) to explore more when the agent does have enough experience while exploiting more as the agent gains experience. Their model is trained by running 20,000 episodes. Human raters are asked to rate which agent (agenda-based or RL-based) performed better by providing the negotiation transcripts. According to their results, the RL-based agent outperformed the agenda-based agent. Unlike our environment, their environment is dynamic; the deadline and agents may change during the negotiation session. In contrast, the deadline does not change arbitrarily and utility values of negotiated issues do not change during the negotiation in our framework. While agents make complete offers aiming to find an agreement on all issues at a time, their agent negotiates issue by issue.

Zou et al. [30] integrate genetic algorithm and reinforcement learning to determine an optimal strategy in negotiation. Their motivation for applying this kind of technique is negotiating with uncertainty about the opponent. They state that their technique achieves better results in terms of efficiency, fairness, and strategy convergence. They also mention that their approach achieves higher reward, shorter negotiation time, and lower degree of greediness compared to the classical evolutionary models. While an evolutionary RL approach is adopted in order to determine the optimal negotiation strategy in that study, we use DQN to learn what and when to accept during the negotiation.

Kröhling et al. [31] address the problem of determining a negotiation strategy during negotiation. They introduce a conceptual entity as an oracle that can be queried by its agent. The agent tells the current context during the negotiation and the oracle estimates the utility value of each strategy using the Q function. While they aim to learn which strategy to use during the negotiation, we focus on learning when to accept the opponent's offer.

Rodriguez et al. [32] study bilateral negotiations for electricity market energy contracts. They introduce a context aware Q-learning approach for energy contracts. In that work, the negotiation issues are the amount of energy and its price. The context in their study refers to the factors influential in the contract prices such as current date and time and amount of transacted power in the electricity market. By using context-aware Q-learning, their agent estimates the values of contracts. Therefore, their study focuses on prenegotiation (i.e. deciding on what contract to negotiate). On the other hand, our study focuses on developing an RL-based acceptance strategy for metanegotiations in which the issues of negotiations are fixed and the agents negotiate on these issues to maximize their utilities.

Sunder et al. [33] develop an agent that negotiates on contracts in industrial scenarios. They use a policy-based RL approach, namely REINFORCE, in order to learn what to bid in a bilateral negotiation setting. Different from the aforementioned approaches, the agent learns the content of its next offer rather than the utility of its next offer. While they employ a policy-based RL approach, we adopted a value-based RL approach (Q-learning). Our work is complementary to their work, since they aim to learn what to bid while we focus on learning when to accept.

In order to highlight the major differences between related work and the proposed approach, a comparison matrix is presented in the following table. While our work proposes using RL for the acceptance strategy, other works using RL in automated negotiations mostly focus on learning a bidding strategy or negotiation strategy. While some works adopt a policy gradient approach such as REINFORCE, most of the works use a value-based approach, particularly Q-learning, as we do. To the best of our knowledge, our work is the first using deep RL aiming to learn when and what to accept in negotiations.

Table 1. Comparison matrix of RL-based approaches in automated negotiation.

Work	**What to learn**	**Method**	**Issue**
Zou	To pick which concession strategy	Evolutionary RL	Multi
Papangelis	To decide which action to take	Q-learning with function approximation	Multi
Kröhling	To determine a negotiation strategy	Tabular Q-learning	Single
Rodriguez	To decide on what domain to negotiate	Tabular Q-learning	Multi
Sunder	To determine what to bid	REINFORCE	Multi
Bakker	To determine what to bid	Tabular Q learning	Multi
Ours	**To decide when and what to accept**	**DQN**	**Multi**

5. Conclusion

This paper proposes a novel acceptance strategy model for bilateral negotiations based on deep RL. This model can be used as an acceptance strategy module under the BOA framework in the GENIUS environment. Our approach could successfully result in a model that can perform well for negotiations with various agents in different domains with comparable results in the test session. Compared to other studies on automated negotiations, our main contribution is using a deep RL approach for learning when the agent should accept its opponent's offer. Our experiment results showed that the RL acceptance strategy performs at least as well as the AC-next strategy, which is the state of the art acceptance strategy in automated negotiations.

In future work, it would be interesting to design models for a bidding strategy that outperforms the existing RL-based bidding strategies. Another idea is the integration of the proposed acceptance policy in a human-agent negotiation environment.

Acknowledgments

We would like to sincerely thank Dr Melih Kandemir, and Artificial Intelligence Lab members at Özyeğin University for their valuable feedback on our work. We appreciate the anonymous reviewers' constructive comments on our work.

Razeghi and Yavuz worked collaboratively on this paper under the supervision of Dr. Reyhan Aydoan.

References

[1] Jennings NR, Faratin P, Lumiscio AR. Automated negotiation: prospects, methods and challenges. Group Decision and Negotiation 2001; 10 (2): 199-215. doi: 10.1023/A:1008746126376

[2] Barslaag T, Gerding EH, Aydoan R, Schraefel MC. Optimal negotiation decision functions in time-sensitive domains. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT); Singapore; 2015. pp. 190-197.

[3] Sánchez-Anguix V, Julián V, Botti V, García-Fornes A. Studying the impact of negotiation environments on negotiation teams' performance. Information Sciences 2013; 219 (1): 17-40. doi: 10.1016/j.ins.2012.07.017

[4] Sanchez-Anguix V, Aydoan R, Julian V, Jonker CM. Intra-team strategies for teams negotiating against competitor, matchers, and conceders. In: Marsa-Maestre I, Lopez-Carmona M, Ito T, Zhang M, Bai Q et al (editors). Novel Insights in Agent-based Complex Automated Negotiation. Japan: Springer, 2013, pp. 3-22.

[5] Tunal O, Aydoan R, Sanchez-Anguix V. Rethinking frequency opponent modeling in automated negotiation. In: International Conference on Principles and Practice of Multi-Agent Systems; Nice, France; 2017. pp. 263-279.

[6] Aydoan R, Yolum P. Ontology-based learning for negotiation. In: IEEE/WIC/ACM Inter-national Joint Conference on Web Intelligence and Intelligent Agent Technology; Milan, Italy; 2009. pp. 177-184.

[7] Baarslag T, Hendrikx MJC, Jonker CM, Hindriks KV. Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. Autonomous Agents and Multi-Agent Systems 2016; 30 (1): 849-898. doi: 10.1007/s10458-015-9309-1

[8] Aydoan R, Festen D, Hindriks KV, Jonker CM. Alternating offers protocols for multilateral negotiation. In: Fujita K, Bai Q, Ito T, Zhang M, Ren F, Aydoan R, Hadfi R (editors). Modern Approaches to Agent-based Complex Automated Negotiation, USA: Springer, 2017, pp. 153-167.

[9] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.

[10] Baarslag T, Hindriks KV, Jonker CM. Effective acceptance conditions in real-time automated negotiation. Decision Support Systems 2014; 60 (1):68-77. doi: 10.1016/j.dss.2013.05.021

[11] Cai H, Ren K, Zhang W, Malialis K, Wang J et al. Real-time bidding by reinforcement learning in display advertising. In: The Tenth ACM International Conference on Web Search and Data Mining; Cambridge, UK; 2017. pp. 661-670.

[12] Borissov N, Anandasivam A, Wirström N, Neumann D. Rational bidding using reinforcement learning. In: Altmann J, Neumann D, Fahringer T (editors). Germany: Springer, 2008, pp. 73-88.

[13] Hindriks KV, Jonker CM, Kraus S, Lin R, Tykhonov D. Genius: negotiation environment for heterogeneous agents. In: The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2; Budapest, Hungary; 2009. pp. 1397-1398.

[14] Baarslag T, Hindriks KV, Hendrikx MJC, Dirkzwager A, Jonker CM. Decoupling negotiating agents to explore the space of negotiation strategies. In: Marsa-Maestre I, Lopez-Carmona MA, Ito T, Zhang M, Bai Q et. al. (editors). Novel Insights in Agent-based Complex Automated Negotiation, Japan: Springer, 2014, pp. 61-83.

[15] Pomerol J, Barba-Romero S. Multicriterion Decision in Management: Principles and Practice. USA: Springer, 2000.

[16] Raiffa H. The Art and Science of Negotiation. Cambridge, MA, USA: Harvard University Press, 1982.

[17] Adar MB, Sofy N, Elimelech A. Gahboninho: Strategy for balancing pressure and compromise in automated negotiation. In: Ito T, Zhang M, Robu V, Matsuo T (editors). Complex Automated Negotiations: Theories, Models, and Software Competitions, Germany: Springer, 2013, pp. 205-208.

[18] Kawaguchi S, Fujita K, Ito T. Agent K: Compromising strategy based on estimated maximum utility for automated negotiating agents. In: Ito T, Zhang M, Robu V, Matsuo T (editors). New Trends in Agent-Based Complex Automated Negotiations. Germany: Springer, 2012, pp. 137-144.

[19] Last N.G. Agent smith: Opponent model estimation in bilateral multi-issue negotiation. In: Ito T, Zhang M, Robu V, Matsuo T (editors). New Trends in Agent-based Complex Automated Negotiations. Germany: Springer, 2012, pp. 167-174.

[20] Baarslag T, Hindriks KV, Jonker MC, Kraus S, Lin R. The first automated negotiating agents competition (ANAC 2010). In: Ito T, Zhang M, Robu V, Matsuo T (editors). New Trends in agent-based complex automated negotiations. Germany: Springer, 2012, pp. 113-135.

[21] An B, Lesser V. Yushu: A heuristic-based agent for automated negotiating competition. In: Ito T, Zhang M, Robu V, Matsuo T (editors). New Trends in Agent-Based Complex Automated Negotiations. Germany: Springer, 2012, pp. 145-149.

[22] Ito T, Zhang M, Robu V, Fatima S, Matsuo T. New Trends in Agent-based Complex Automated Negotiations. Germany: Springer, 2011.

[23] Williams CR, Robu V, Gerding EH, Jennings NR. Iamhaggler: A negotiation agent for complex environments. In: Ito T, Zhang M, Robu V, Matsuo T (editors). New Trends in Agent-based Complex Automated Negotiations. Germany: Springer, 2012, pp. 151-158.

[24] Khosravimehr Z, Nassiri-Mofakham F. Pars agent: Hybrid time-dependent, random and frequency-based bidding and acceptance strategies in multilateral negotiations. In: Fujita K, Bai Q, Ito T, Zhang M, Ren F et al (editors). Modern Approaches to Agent-based Complex Automated Negotiation. Germany: Springer, 2017, pp. 175-183.

[25] Jonker CM, Aydoan R, Baarslag T, Fujita K, Ito T et al. Automated negotiating agents competition (ANAC). In: The Thirty-First AAAI Conference on Artificial Intelligence; San Francisco, California USA; 2017. pp. 5070-5072.

[26] Fujita K, Aydoan R, Baarslag T, Hindriks KV, Ito T et al. The sixth automated negotiating agents competition (ANAC 2015). In: Fujita K, Bai Q, Ito T, Zhang M, Ren F et al (editors). Modern Approaches to Agent-based Complex Automated Negotiation. Germany: Springer, 2017, pp. 139-151.

[27] Bakker J, Hammond A, Bloembergen D, Baarslag T. RLBOA: A modular reinforcement learning framework for autonomous negotiating agents. In: The 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'19); Budapest, Hungary; 2019. pp. 260-268.

[28] Papangelis A, Georgila K. Reinforcement learning of multi-issue negotiation dialogue policies. In: The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue; Prague, Czech Republic; 2015. pp. 154-158.

[29] Rudnicky A, Xu W. An agenda-based dialog management architecture for spoken language systems. In: IEEE Automatic Speech Recognition and Understanding Workshop; Sentosa, Singapore; 1999. pp. 1-17

[30] Zou Y, Zhan W, Shao Y. Evolution with reinforcement learning in negotiation. PLOS One 2014; 9 (7): 1-7. doi: 10.1371/journal.pone.0102840

[31] Kröhling D, Hernández F, Martínez E, Chiotti OJA. The importance of context- dependent learning in negotiation agents. Inteligencia Artificial 2018; 22 (63): 135-149. doi: 10.4114/intartif.vol22iss63pp135-149

[32] Rodriguez-Fernandez J, Pinto T, Silva F, Praça I, Vale Z et al. Context aware Q-learning-based model for decision support in the negotiation of energy contracts. International Journal of Electrical Power & Energy Systems 2019; 104: 489-501. doi: 10.1016/j.ijepes.2018.06.050

[33] Sunder V, Vig L, Chatterjee A, Shroff G. Prosocial or selfish? agents with different behaviors for contract negotiation using reinforcement learning. In: Ito T, Aydoğan R, Zhang M (editors). In Advances in Automated Negotiations. Germany: Springer, 2020, pp. 69-88.