Turkish Journal of Electrical Engineering & Computer Sciences

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

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 1695 – 1714

doi:10.55730/1300-0632.3899

Actor-critic reinforcement learning for bidding in bilateral negotiation

Furkan ARSLAN[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: 25.08.2021 • **Accepted/Published Online:** 26.04.2022 • **Final Version:** 22.07.2022

Abstract: Designing an effective and intelligent bidding strategy is one of the most compelling research challenges in automated negotiation, where software agents negotiate with each other to find a mutual agreement when there is a conflict of interests. Instead of designing a hand-crafted decision-making module, this work proposes a novel bidding strategy adopting an actor-critic reinforcement learning approach, which learns what to offer in a bilateral negotiation. An entropy reinforcement learning framework called Soft Actor-Critic (SAC) is applied to the bidding problem, and a self-play approach is employed to train the model. Our model learns to produce the target utility of the coming offer based on previous offer exchanges and remaining time. Furthermore, an imitation learning approach called behavior cloning is adopted to speed up the learning process. Also, a novel reward function is introduced that does take not only the agent's own utility but also the opponent's utility at the end of the negotiation. The developed agent is empirically evaluated. Thus, a large number of negotiation sessions are run against a variety of opponents selected in different domains varying in size and opposition. The agent's performance is compared with its opponents and the performance of the baseline agents negotiating with the same opponents. The empirical results show that our agent successfully negotiates against challenging opponents in different negotiation scenarios without requiring any former information about the opponent or domain in advance. Furthermore, it achieves better results than the baseline agents regarding the received utility at the end of the successful negotiations.

Key words: Deep reinforcement learning, entropy reinforcement learning, imitation learning, automated bilateral negotiation, bidding strategy, multi-agent systems

1. Introduction

Automated agreement technologies provide promising solutions to a variety of problems such as resource allocation [1] and path planning [2], and have a vast range of applications including electronic commerce [3], coordination in robotics [4], privacy management for information sharing systems [5], and so on. Thus, there is increasing attention to the field of automated negotiation in multi-agent systems. Researchers have designed various negotiation strategies [6–9] so far in order to resolve conflicts on the underlying problems. As automated negotiation involves mostly exchanging offers by the negotiating parties under a particular negotiation protocol, one of the most compelling research challenges is to design a decision-making module to determine what to offer, namely bidding strategy.

To date, researchers aim to design effective bidding strategies to determine the most appropriate offer to make at a particular time so as to lead their agent to beneficial negotiation outcomes. Consequently, various bidding strategies have been proposed so far in the literature. While some researchers focus on designing agents

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

that are capable of negotiating with others only on a particular domain [10], the majority focuses on designing *generic* agents that can negotiate in different domains, including a varying number of issues and having different sizes of outcome space. It is not a straightforward task since each negotiation scenario/problem involves different dynamics. That is, the complexity of the problem may vary in terms of the size of the outcome space (e.g., a few possible outcomes versus large-scaled outcome space) and conflict degree (e.g., collaborative versus competitive scenario). Moreover, there is uncertainty about the opponent's preferences and strategy. While some opponents are collaborative, others might act selfishly. Therefore, the designed agents should handle this uncertainty and adapt their behavior according to their environment.

The classical techniques used in bilateral negotiations are mostly heuristic-based approaches such as time-dependent concession strategies or behavior-based concession strategies that reciprocate the opponent's behavior during the negotiation [11]. This process may require learning the opponent's preferences. Consequently, a vast number of studies have focused on opponent modeling using some heuristic approaches [12, 13, 43] and adopting some machine learning techniques such as Bayesian learning and some concept-based learning methods [11, 41]. On the one hand, the aforementioned bidding strategies are mostly predefined. That is, they have a limited capacity for behavior adaptation in their ongoing negotiation (e.g., speed of concession, the level of mimicking the opponent's behavior). On the other hand, a negotiating agent can evolve and improve negotiation skills over time based on previous negotiation experiences. Reinforcement Learning (RL) would be a promising approach to learn what actions to take during the negotiation to maximize the expected future negotiation outcome. Like a chess champion, an agent may practice and learn how to negotiate repeatedly to maximize its gain in its future negotiations. Therefore, this study mainly focuses on how an RL-based negotiation strategy can be modeled for bilateral negotiations so that the agent can observe different behaviors and get insights into the potential consequences of its actions in a given state.

Accordingly, there are some recent attempts to adopt RL in the context of negotiation. For instance, Sunder *et al.* apply RL in order to generate the content of the current offer [14]. However, their model is limited to the negotiation domain used in training. For other domains with different issues and values, the model needs to be trained on those domains again. Moreover, Bakker *et al.* use the Q-learning to calculate the desired target utility range so that agents can generate a suitable offer within this range by using opponent modeling. On the one hand, this approach is domain-independent so that the trained model could be used in other negotiation domains. On the other hand, they consider a discrete action space limiting the agent's moves. A more sophisticated deep RL approach could be developed by considering continuous state and action spaces. Most recent works [15, 16] adopt actor-critic RL approaches supporting continuous action space to find optimal target utility. Those RL models are trained by negotiating with a particular set of agents in some scenarios. That is, training is performed by exploring a limited space based on those opponents' strategies. There is room for improvement by adopting a self-learning and behavior cloning [17] approach.

Accordingly, this article proposes a novel bidding strategy using an actor-critic approach namely Soft Actor-Critic. The proposed strategy, namely *SAC Agent* uses the power of maximum entropy Reinforcement Learning to learn how to negotiate independently of scenario and opponent from scratch. We intend to find the most appropriate target utility to make offers without requiring any opponent model by considering continuous state and action spaces. Unlike other approaches, we adopt a self-learning approach in which our agent negotiates with itself to explore a vast state space. Since self-learning requires a large amount of data and training to find the optimal policy, we create a dataset from the negotiation transactions of successful agents to speed up the learning process and adopt a behavior cloning approach. The proposed approach uses its own experiences as

well as demonstration data from the experts. Our experimental results show that our agent can learn bidding over time and even outperform most of the top-performing state-of-the-art agents.

The rest of this paper is organized as follows: Section 2 briefly discusses the related works. Section 3 provides the necessary background for negotiation and RL while Section 4 explains the proposed RL-based bidding strategy. Section 5 describes the experimental setup, demonstrates the achieved results in both training and testing negotiation sessions, and also includes an explanation about the methodologies used for the generalization of the model. Finally, Section 6 concludes the paper and discusses the future work.

2. Related work

Along with the successes of RL, it has also started to be used in negotiation in recent years. [29, 30] have shown the success of the Q-learning algorithm, which is the most widely used RL algorithm in price negotiation. Apart from the price negotiation, it has also been used in dynamic pricing selection [31] and agenda-based negotiation systems [32]. RL has shown promising results not only in issue-based negotiations but also in dialog-based negotiations [33]. Some researchers focus on learning the content of the offer itself. For instance, Sunder *et al.* analyze the behavior of RL negotiation agents that have prosocial and selfish behavior trained by playing with each other in the domain of contract negotiation [14]. In this consideration, they create a communication protocol in which each issue values of the offer are represented in binary formation. Consequently, they aim to learn what to offer in binary format. This approach differs from other RL approaches that aim to learn the target utility rather than the content of the offer directly. They can adjust the behavior of the RL agents (i.e. prosocial and selfish) by tweaking their reward function. Their experimental results showed that agents could learn to act as expected with respect to the desired behavior. On the other hand, their approach is not fully independent of the negotiation domain (i.e. issues and possible values). To be able to negotiate in another domain, agents need to be trained in that particular domain. Other studies, including ours, present a more generic approach in contrast to this study. The learned models are applicable for unknown domains in our case. Kröhling, Chiotti, and Martinez claim that using the bargaining context and important external variables can give the agent a considerable advantage, especially when modeling the opponent's strategy, behaviors, and assumptions [34]. On the importance of all this contextual information on bargaining, they create an agreement environment for automated negotiation that includes both contextual variables and context-sensitive intermediaries. They propose a new bargaining agent model based on RL that takes contextual variables into account within this new negotiation environment. This agent, named Strawberry, queries the available risk and necessity information from the environment, evaluates, and uses it.

In recent years, different RL approaches have been proposed in automated negotiation. For example, Bakker *et al.* use tabular Q-learning to learn when and to what extent it should concede during the negotiation [24]. In this approach, the agent decides whether to move one bin down from the highest bin (i.e. 0.9-1) to stay in the same bin or to go up once. This approach uses discrete state and action space to narrow down its exploration space. However, this approach would not give a fine-grained target utility estimation. In addition, it requires an opponent modeling approach to find the appropriate offer whose utility falls within the target utility range. In contrast, our approach aims to learn the target utility rather than a utility range by adopting an actor-critic reinforcement learning approach. Moreover, Razeghi *et al.* propose to use a Deep RL approach to learn when to accept the opponent's offer in automated bilateral negotiation [35]. Their experimental results show that the proposed RL-based acceptance strategy performs as well as the AC-next strategy, the most widely-used strategy. Our work is complementary. While our work focuses on what to bid, that study aims to

learn when to accept its opponent's offer. Bagga, Paoletti, and Alrayes *et al.* introduce an RL approach for both acceptance and bidding strategies. In this work, a model-free actor-critic agent (DDPG) is modeled for performing concurrent negotiations which successfully adopt different e-market settings (such as e-bay). Their model uses a tabular Q-learning approach to learn whether or not to accept the opponent's current offer and the actor-critic RL approach to generate continuous target utility of the offer in the bidding phase [15]. The main disadvantage of this approach is that training two RL agents simultaneously can provide inconsistency because the result of one agent affects the other agent's result. For example, if the Q agent converges to a wrong policy, it can mislead the policy improvement of the DDPG agent. Therefore, we adopt a constant acceptance strategy in our work.

Sengupta *et al.* recently propose an actor-critic approach for bidding and a mechanism for selecting the most appropriate trained RL agents among a set of RL agents [16]. They train the proposed RL agents by making them negotiate with a fixed set of opponents such as Boulware, Nice TFT, and Conceder agents in several negotiation scenarios. They also use a SAC bidding strategy with a fixed acceptance strategy similar to our approach. That work advocates training different RL agents by training with different opponents and selecting an RL agent against the current by predicting which RL agent would be the best for that opponent. Furthermore, based on its prediction, the negotiator agent may switch or combine the strategies to use against the current opponent. One of the main differences between their approach and ours is that we only create a single instance of RL agent, which learns how to bid by negotiating with itself instead of a fixed opponent, and we adopt a behavior cloning approach to speed up the learning process. In their approach, they train multiple RL agents and adopt a mechanism selection approach or a mixture of expert approaches as proposed in [36]. Adaptive switching and training each agent separately increases the complexity of the learning process. We reduce the complexity by using the actor-critic approach with behavior cloning and self-learning. Thanks to this combined approach, we achieved successful results against our competitors in different domains without retraining. In addition, the new reward function we have introduced is one of the essential contributions that affect our agent's success. In our reward function, we consider the utility of the opponent as well as ours, while they only consider the agent's own utility in their reward function. We conclude our related work with a comparison matrix in which most related RL approaches are compared according to a number of criteria as seen in Table 1.

Table 1. Functionality comparison of the existing work.

	Lewis et al. [33]	**Jin et al.** [37]	**Sunder et al.** [14]	**Bakker et al.** [24]	**D. Kröhling et al.** [34]	**Razeghi et al.** [35]	**Bagga et al.** [15]	**Sengupta et al.** [16]	**SAC Agent**
Domain independent	√	×	×	√	×	√	√	√	√
Acceptance Criteria	×	ACnext	ACnext	ACnext	ACnext	RL Agent	RL Agent	ACcombi	ACnext
RL algorithm	Reinforce with baseline	Deep Deterministic Policy Gradient	Reinforce with baseline	Tabular Q Learning	Tabular Q Learning	Deep Q Learning	Tabular Q Learning / DDPG	SAC	SAC
Continues State/Action	×	√	√	×	×	×	√	√	√
Self learning	×	×	√	×	×	×	×	×	√
Learning Bid generation	√	√	√	√	√	×	√	√	√
Learning Acceptance	×	×	×	×	×	√	√	×	×
Meta agent	×	×	√	×	×	×	×	√	×

3. Background

In this section, we provide the necessary background on automated negotiation (Section 3.1) and Reinforcement Learning (Section 3.2). The abbreviations used in this paper are summarized in Table 2.

Table 2. Glossary of abbreviations.

Abbreviations	**Meaning**
PE	Policy evaluation
PI	Policy improvement
MDP	Markov decision process
ReLU	Rectified Linear Unit activation function
TD	Temporary difference
RL	Reinforcement learning
BC	Behavior cloning
SAC	Soft actor critic
AC-Next	Acceptance condition next
GENIUS	General Environment for Negotiation with Intelligent Multi Purpose Usage Simulation
ANAC	The International Automated Negotiating Agents Competition
RLBOA	A Modular RL Framework for Autonomous Negotiating Agents

3.1. Automated negotiation

In general, negotiations are about a finite set of n issues $\mathcal{I} = \{1, 2, \ldots, n\}$. Each issue $i \in \mathcal{I}$ has a range $\mathcal{D}_i$ of possible instantiations. An outcome is a complete assignment to the set of issues, i.e. an element of Cartesian product of the ranges of instantiations per issue. Formally, the set of all possible outcomes is defined as $\mathcal{O} = \mathcal{D}_1$ X $\mathcal{D}_2$ X... X $\mathcal{D}_n$. An offer is represented by o, while $\mathcal{O}^*$ represents the set of all possible offers in the negotiation domain. The agents' preferences are represented by means of linear additive utility functions as shown in Equation 1 where w_i represents the importance of the negotiation issue i for the agent, o_i represents the value for issue i in offer o, and $V_i(.)$ is the valuation function for issue i, which returns the desirability of the issue value. Without losing generality, it is assumed that $\sum_{i \in \mathcal{I}} w_i = 1$ and the domain of $V_i(.)$ is in the range of [0,1] for any i.

$$\mathcal{U}(o) = \sum_{i \in \mathcal{I}} w_i \times V_i(o_i) \tag{1}$$

The interaction among agents during an automated negotiation is governed by the *stacked alternating offers protocol* (SAOP) [18]. In SAOP, agents have a shared deadline to reach an agreement, and one of the agents initiates the negotiation with an offer (first round). In each turn, the agent receiving an offer can *accept* the current offer, make a *counteroffer*, or *end* the negotiation without an agreement. Turn-taking continues until success (agreement is reached: all agents accept) or failure (no agreement: any agent ends the negotiation or the deadline is reached). An agent negotiates by reasoning about its user's preference profile, often represented via a utility function, as well as taking its opponent's offers into account. In general, an agent does not know its opponent's preferences or its opponent's negotiation strategies.

3.2. Reinforcement learning (RL)

Reinforcement learning is a goal-oriented optimization technique that aims to learn an *optimal policy* by interacting with the environment through trial and error [19]. An optimal policy, which is a function mapping states to actions $\pi* : S \rightarrow A$, specifies what actions to be taken in the underlying states to maximize future expected rewards. We consider the Markov decision process (MDP) framework for selecting optimal actions to maximize rewards over discrete time steps in a fully observable environment E. At every time step t, an agent is in a state s_t, takes an action $a_t = \pi(S_t)$, receives an immediate reward r_t, and E reaches to state s_{t+1} with a transaction probability $P(s_{t+1}|s_t, a_t)$. The agent keeps this interaction in its experience buffer in the form of $< s_t, a_t, r_t, s_{t+1}, d_t >$ where d_t holds to whether the interaction at t is an end state. Accordingly, it uses this information while updating its policy.

In episodic tasks, interaction with the environment breaks into subsequences. The environment restarts after the *terminal state* T or *final step*. That is, a new subsequence starts in each episode. Note that an episode corresponds to the whole steps of a game from the beginning to the end. In our context, it is a negotiation session including all offer exchanges in a single session ending with a success or failure. During the learning process, agents aim to maximize the return of the underlying episodes. In other words, the main objective of the agent is to maximize the discounted sum of future rewards called expected return shown in Equation 2 where γ is a discount factor for future rewards, which is between 0 and 1. In the negotiation context, that corresponds to maximizing the future utility of the negotiation outcomes.

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + r_{t+4} + \cdots + r_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{2}$$

RL uses two important function concepts derived from the Bellman equations. The first is the value function estimating the expected future reward of being in a given state for the agent – how desired that state would be for the agent. Since the future rewards depend on what actions the agent will take, the value function is estimated concerning the agent's policy, which determines how it will act. Accordingly, as shown in Equation 3 [19], value functions are defined with respect to policies (v_π) where $E_\pi[]$ denotes the expected value of a random variable given that the agent follows policy π, and t indicates a time step. The second concept, the action-value function called Q function is similarly defined in Equation 4.

$$v_\pi(s) = E_\pi[G_t|\mathcal{S}_t = s] = E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k|\mathcal{S}_t = s\right], forall s \in \mathrm{S} \tag{3}$$

$$q_\pi(s, a) = E_\pi[G_t|\mathcal{S}_t = s, \mathcal{A}_t = a] = E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k|\mathcal{S}_t = s, \mathcal{A}_t = a\right] \tag{4}$$

RL algorithms struggle to find convergence in continuous and large-scaled RL problems while optimizing the value function and policy separately. The actor-critic approaches like SAC handle this problem by performing concurrent updates of those functions. In actor-critic RL approaches, the policy is considered as the actor and the value function as the critic. In general actor-critic algorithms are based on the policy iteration which alternates between *Policy Evaluation (PE)* as in shown in Equation 5 and *Policy Improvement (PI)* as shown in Equation 6 [19], which is based on the greedy policy. While the former is about computing the value function

for a policy, the latter uses the value function to find a better policy [20].

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right] \tag{5}$$

$$v_{\pi'}(s) = \max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_{\pi'}(s')\right] \tag{6}$$

4. Proposed bidding strategy: SAC-Agent

One of the essential decisions regarding the design of the RL model is whether we should adopt a *discrete* or *continuous* action space. In discrete action space, actions can be considered as target utility bins in automated negotiation (i.e. utility ranges such as a utility between 0 and 0.1, between 0.1 and 0.2, and so on). In that case, the agent should select one of the offers whose utility fits in that range; however, there might be multiple offers with varying utilities in that range (e.g., 0.155, 0.166, 0.170, etc.). Even the slightest utility change may significantly affect the negotiation outcome since it affects which offer to be made in the current round. In the discrete space approach, these details are lost. Therefore, we adopt a continuous action space in which an action corresponds to the utility of the offer rather than the utility range. This would lead the agent to make its decisions more precisely.

Besides, in bilateral negotiations, negotiating agents do not know how their opponent will respond to their current offer (i.e. not know the explicit behavior of the opponents). Therefore, it would be more appropriate to use a model-free approach such as TD-based methods combining ideas from Monte Carlo and Dynamic Programming [19]. Furthermore, TD-based approaches are capable of learning continuous action spaces. Due to the reasons mentioned above, we adopt the Soft Actor-Critic (SAC) [20], the off-policy actor-critic algorithm based on the maximum entropy RL approach to create a generic agent capable of negotiating with any opponents in any domain. Our model aims to learn an optimal policy by mapping given negotiation state into a valid action – a target utility of the offer to be made by our agent. Although the learned policy does not guarantee the optimal outcome, the SAC algorithm, which uses an off-policy maximum entropy deep learning approach, tends to converge to the optimal policy [42]. The general learning structure of our *SAC Agent* is demonstrated in Figure 1. The main intuition is to make our agent negotiate with itself repeatedly to explore finite continuous states over time so that it learns what action to take among continuous action space from scratch. Since this process requires many iterations, we adopt imitation learning called behavior cloning to speed up the learning process. That is, the agent uses an additional dataset called *demonstration data*, which is composed of negotiation transactions from experts.

When the agent negotiates with itself repeatedly, it may have different negotiation experiences at each session since its behavior would change over time due to its updated policy. In addition, it would be better to explore different negotiation scenarios (e.g., adopting a different preference profile) to increase the diversity of negotiation environments. To memorize all experiences that the agent gains so far, an *experience replay buffer* [21] is used. An experience replay buffer can be considered as accumulated negotiation traces from the previous and current negotiations.

On-policy learning approaches used in RL methods are data-hungry methods as they require a new sample in each learning step. Off-policy learning used by the SAC method aims to use past experiences; that is why it is a more data-efficient approach than on-policy learning. The combination of domain complexity, continuous

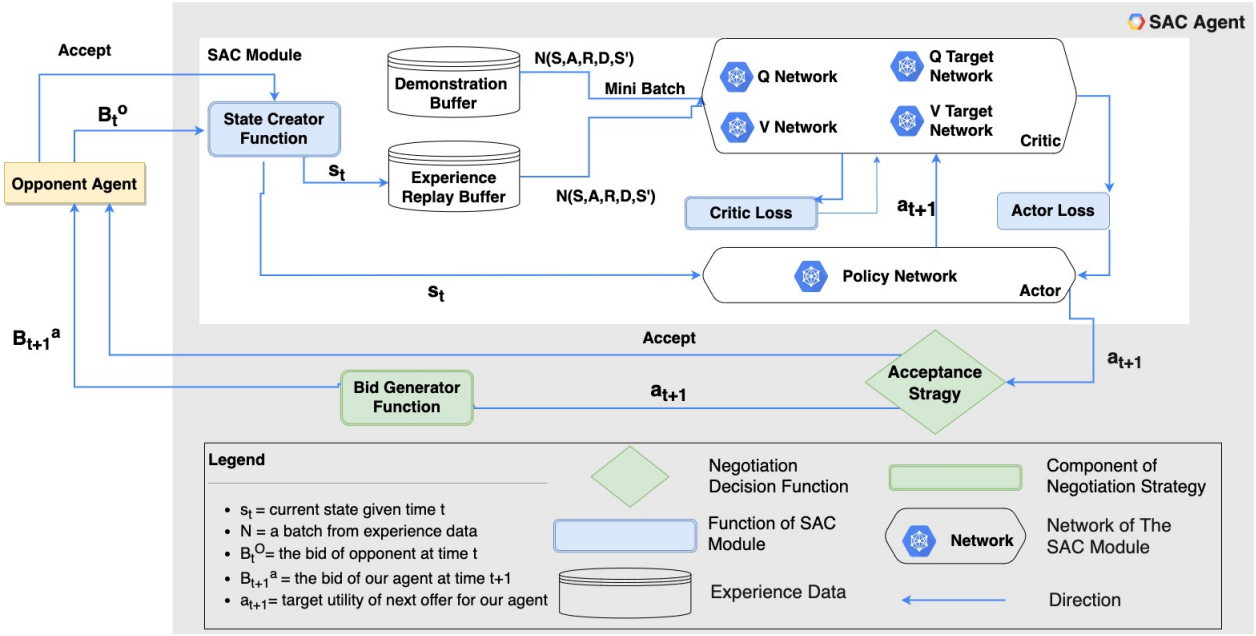

Figure 1. Overall structure of the proposed SAC Agent.

action, state spaces, and nonlinear function approximation with neural networks presents a significant challenge for stability and convergence even for off-policy learning RL. Thanks to the entropy maximization approach, the SAC tends to be more stable than other off-policy approaches, independent of hyperparameters. However, it still requires much training and has convergence problems due to the sizeable continuous action and state space and the low and late arrival of the reward signal according to our observations in the negotiation environment. Therefore, we use the behavior cloning approach by using the data generated from top bilateral negotiating agents in the International Automated Negotiating Agents Competition (ANAC) [9] to reduce the training time of the *SAC Agent* and support its converge. The demonstration replay buffers store same format as $< s_t, a_t, r_t, s_{t+1}, d_t >$. In the proposed approach, the *SAC Agent* uses both demonstration and replay data while performing actor and critic updates for each mini-batch.

SAC combines off-policy actor-critic training with a scholastic actor and adopts an entropy maximization objective [20]. It aims to maximize both the expected return and entropy of the actor. The entropy controls the randomness in the policy. Increasing entropy enforces more exploration and consequently leads to success in terms of exploration and exploitation. In addition, the trade-off between expected return and entropy makes policy more stable and robust [20]. As defined in [20], Equation 7 shows the policy objective with entropy maximization where $\mathcal{H}$ denotes to entropy function and α determines relative importance of the entropy. We use the same behavior cloning approach with [22]. Equation 8 shows the behavior cloning loss (Equation 8) computed only on the demonstration examples for training actor, which introduced by Ashvin Nair *et al.* [17]. This loss aims to improve the learned policy without going beyond the demonstration data. Considering the gradient applied to the parameters θ_π described as $\lambda_1 \nabla\theta_\pi J - \lambda_2 \nabla\theta_\pi L_{BC}$, the aim is to maximize expected return J and minimize behaviour cloning loss L_{BC}.

$$\pi* = \arg\max_{\pi} \sum_{t} E_{S_t, a_t \sim p_\pi}[r(S_t, a_t) + \alpha\mathcal{H}(\pi(.|S_t))] \tag{7}$$

$$L_{BC} = \sum_{i=1}^{N_D} ||\pi(S_i|\theta_\pi) - a_i||^2 \tag{8}$$

Furthermore, the critic element of the SAC algorithm consisting of Q (action network) and V (value network) networks adopts the dual Q network approach to control value (V) and action value (Q) updates. That is, a target network exists for each corresponding network to balance its output. While value networks are fed only state information, the Q networks are fed with state and action information derived from the mini-batch and actor. Their outputs are used in the loss function calculation of the actor. The actor element of the SAC algorithm only consists of a Gaussian policy network learning the optimal stochastic policy to determine the best action to take at the underlying state. Note that *the TanH activation function* in the output layer produces a continuous numeric value between -1 and 1. Therefore, we needed to map the output of the SAC algorithm into a value between zero and one in order to use those values as target utilities in the GENIUS environment.

Recall that state information is given as an input into the SAC model and the model itself produces the target utility for the opponent's current offer, which is between zero and one. If the target utility is lower or equal to the utility of its opponent's last offer, the agent would prefer to accept the opponent's offer instead of making a counteroffer. This is the most-widely used acceptance criterion in automated negotiation called *AC-Next* [9]. Otherwise, it will make an offer with the target utility. If there is no available offer with that utility in the given scenario, the agent chooses an offer whose utility is the closest to the estimated target utility.

In order to enable the *SAC Agent* to work in all domains without the need for reconfiguration, we adopt a state definition including domain-independent information such as the utility of the opponent's current offer and the utility of our agent's previous offer as well as the current normalized time step. Since the offer history plays a crucial role in detecting the opponent's behavior so that the agent can adapt accordingly, our state definition captures subsequent offer exchanges, corresponding to the N-Stack method in RL. Similar to the Atari-DQN study [23], the observation stack mechanism is used for the agent to access historical information. Instead of the entire history, we focus on only four current history steps since it is informative enough. However, the approach is general enough to adapt to any size of history. As a result, state information is a window composed of the utilities of four offer exchanges with respect to the agent's own preferences and the current normalized time step as shown in Equation 9 where U_i^a and U_i^o denote the utility of the agent's offer (U_i^a, $U_i^o \in [0.0 - 1.0]$) and the opponent's offer at time step i respectively, and $\mu \in [0.1 - 0.9]$ denotes the normalized negotiation time.

$$S_t = (\mu, \cup_{i=t-1}^{t-4}(U_i^a, U_i^o)) \tag{9}$$

As the reward signal of the RL agent, mostly the received utility of the agreement by the agent is considered [15, 24]. However, the utility received by an opponent is as important as the utility obtained by the agent at the end of the negotiation. Let us consider two cases where our agent receives the same utility y and the opponent's utility is x in the first case, whereas it is z in the second case. If the agents only consider its received utility, it will gain the same reward (y) for both cases. However, if z is much higher than x and y, and x is lower than y, we could easily say that the second negotiation is more successful for the agent where it beats its opponent. We could say the other way around for the first negotiation where the opponent beats our agent since it receives higher utility than ours. Therefore, we define a reward function, which takes both agent's and the opponent's utility into account. The proposed reward function is given in Equation 10 where $U^a_{agreement}$ and $U^o_{agreement}$ denote the received utility of the agreement by the agent and the opponent, respectively. If

there is no agreement when the deadline is reached, the agent gets a reward of -1 penalty point.

$$R = \begin{cases} \frac{(U^a_{agreement})^2 * 100}{U^o_{agreement}} & \text{if agreement} \\ -1 & \text{no agreement} \end{cases} \tag{10}$$

5. Evaluation

In order to evaluate the performance of the proposed bidding strategy, called *SAC Agent*, we run negotiation tournaments on a variety of negotiation scenarios in the simulation platform called General Environment for Negotiation with Intelligent Multi-Purpose Usage Simulation (GENIUS) [25]. This platform has been widely used by the negotiation research community since 2010. It provides a benchmark of negotiation scenarios, protocols, and strategies and enables researchers to test their agents against state-of-the-art negotiation agents. It also includes all agents developed for ANAC, which is a yearly organized international competition where participants design negotiating agents in the context of a given research challenge [9]. Since the research focus is designing an effective negotiation strategy, particularly for bilateral negotiations, we only consider the successful agents of the ANAC 2011-2012 [26, 27] in our evaluation. Note that the ANAC competition annually introduces a new challenge for the community (e.g., repeated negotiation in 2013, negotiating with nonlinear utility functions in 2014, multilateral negotiation between 2015-2018, and negotiating with partial preferences in 2019-2020). Thus, agents are designed by taking the underlying assumptions into account. In 2011-2012, the main challenge was designing an effective agent with additive utility functions for bilateral negotiations. Therefore, we only consider the most successful agents from those repositories. The agents designed in other years have different assumptions, making it unfair to compare our agent with them.

In the following sections, we describe the training session of the *SAC Agent* (Section 5.1) and present the experiment results in a test setting where *SAC Agent*, *Random Agent* and *RLBOA Agent* negotiate separately with the opponents consisting of the ANAC 2011 finalist agents (Section 5.2).

5.1. Training session

Each neural network consists of three fully connected layers in our SAC implementation. The TanH activation function is used for the output layer, while the Rectified Linear Unit (ReLU) activation function [39] is employed for the other layers. The grid search [38] is performed to tune the most appropriate configurations for our three-layer neural networks. Accordingly, the number of nodes for each layer is set to 512, 256, and 128. The chosen values for other hyperparameters for the SAC implementations are given in Table 3. Note that most of the hyperparameters are the same as the original implementation of the SAC. For agent implementation, we extended the Pytorch [40] library that includes RL model implementations. Training of the *SAC Agent* and negotiation tournaments were run on a server with Intel Xeon Silver 4214R Processor, 48 GB RAM, and NVidia Geforce Quadro RTX 5000 GPU.

We constitute a demonstration buffer by using negotiation transactions obtained from the negotiations among the ANAC 2012 [27] top agents to adopt the imitation learning approach. To achieve it, we ran in total 512 different negotiation sessions among CHUKAgent, AgentLG, OMACAgent, TheNegotiatorReloaded, and AgentMR on the *party domain* consisting of 9 different profiles resulting in 72 different negotiation scenarios. In the party domain, there are six negotiation issues with varying values (i.e. 4, 4, 4, 4, 3, 4 values for the issues of foods, drinks, location, invitations, music, and clean up respectively), which result in $4^5 * 3 = 3072$ possible

Table 3. The hyperparameters of the SAC implementation.

Parameter	Value	Parameter	Value	Parameter	Value
Gamma	0.99	Experience memory	1.10^6	Q/Value learning rate	3.10^{-4}
Batch size	256	Initial random action	1.10^5	Weight entropy	1.10^{-3}
Tau	5.10^{-3}	Actor learning rate	3.10^{-4}	Entropy learning rate	3.10^{-4}
Lambda1	1.10^{-3}	Lambda2	1/128	Demo batch size	128

outcomes. The selected ANAC 2012 agent negotiates with each other and itself on this domain in a bilateral fashion under the deadline of 100 rounds. As a result of those negotiations, we collected 40618 negotiation transactions from the GENIUS platform, a highly reliable platform for the negotiation community in multi-agent systems. Note that a negotiation transaction corresponds to offer exchanges in a single negotiation round; that is, our agent makes an offer, and the other party responds, and vice versa. This demonstration buffer and experience buffer are used to train the *SAC Agent* by making it negotiate with itself on the *party domain* in total 54000 negotiation sessions. Figure 2 shows the changes of the received utility over those negotiation sessions where each point in the graph denotes the average utility of 250 consecutive negotiation sessions. As can be clearly seen, the agent increases its received utility over time. In addition, when we compare the results of the SAC implementation with and without BC, it was clearly seen that the demonstration data obtained from the experts increased the utility of *SAC Agent*.

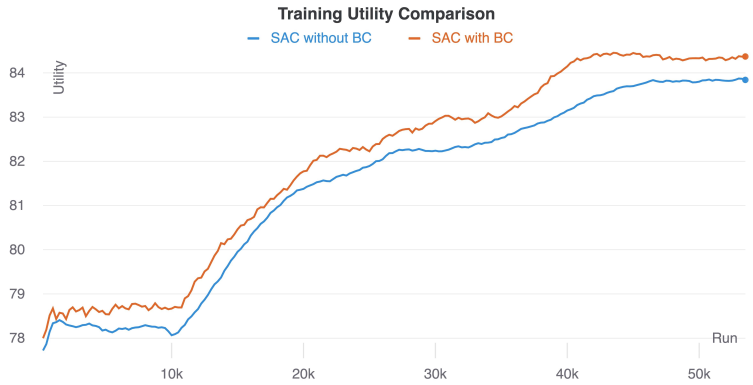

Figure 2. The moving average utility results of training session taken from *SAC* with BC and *SAC* without BC.

5.2. Performance evaluation in test scenarios

In order to demonstrate the generality of our approach, we tested our agents by running a tournament against the ANAC 2011 finalist agents in a variety of scenarios with varying sizes of outcome space with different opposition degrees. Those negotiation scenarios are selected from the ANAC repository and grouped according to the size of their outcome space as shown in Table 4. The opposition corresponds to the distance from the Kalai-Smorodinsky point to the point of maximum utility for both parties as defined in [28]. A high opposition value means the more competitive the domain is. It is worth noting that the proposed *SAC Agent* is trained only on *party* domain so that the agent does not have any negotiation experience with the domains in our test setting.

Table 4. Test scenario information.

Domain	**Plane**	**Itex vs Cypress**	**Airport Site Selection**	**England vs Zimbabwe**	**Grocery**	**Amsterdam**	**Energy (small)**	**Supermarket**
Outcome space	27	180	420	576	1600	3024	15625	98784
Opposition value	0.606	0.431	0.285	0.272	0.191	0.223	0.43	0.347
Outcome class	Small	Small	Small	Small	Medium	Medium	Large	Large
Opposition class	High	High	Medium	Medium	Low	Medium	High	High
Year	2013	2012	2012	2012	2011	2011	2012	2012

In these tournaments, each agent only knows their own preferences. As baseline agents, we consider the *Random Agent* to show that our agent can learn how to offer and consequently can perform better than an agent making random offers, and *RLBOA* to show that our approach can even negotiate better than another existing RL-based approach available in GENIUS environment. *Random Agent* is a baseline agent that generates offers randomly at each negotiation step while *RLBOA* [24] uses the tabular Q-learning to learn the bidding strategy by discretizing state and action space. It uses an opponent modeling to produce appropriate within selection target utility interval.

To sum up, our agent *SAC Agent*, *Random Agent* and *RLBOA Agent* negotiate with the ANAC 2011 [26] finalist agents 10 times for each scenario. The deadline for these negotiation sessions is set to 100 rounds. When they have an agreement, each party receives the utility of the negotiation outcome according to their own preference profiles. That is, agents may end up with different utility scores. As a result of the deal, whichever of the two agents receives the highest utility is deemed to be more successful for that negotiation session. When the negotiation fails (i.e. agents do not reach an agreement when the deadline is reached), both agents receive the utility of zero.

In this part, the performance of the *SAC Agent* is evaluated elaborately by comparing its performance with other agents with respect to the following metrics:

- ***Agreement rate***: It denotes the rate of the number of successful negotiations ending with an agreement over the number of all negotiations.
- ***Received utility***: It is the utility of the negotiation outcome with respect to the preference profile of the agent. If there is no agreement, the agents receive a utility of zero. Note that the utilities range between zero and one in GENIUS. For better visualization, we multiply them by 100 so that the range of the utilities is between zero and 100 in our case.
- ***Winning rate***: We calculate the winning rate as the rate of the number of times that the agent beats its opponent over the total number of negotiations. We consider an agent beating its opponent when it receives a higher utility than or the same utility as the utility obtained by its opponent ($U^a(o) \geq U^0(o)$).

We first evaluate the performance of our agent against its ANAC 2011 opponents. The *SAC Agent* negotiates with each ANAC 2011 finalist agent 10 times on the scenarios listed in Table 4 separately. That is, our agent performed 160 negotiations with each opponent (= 2 profiles * 8 eight domains * 10 repetition). In total, 1280 negotiations took place between our agent and the ANAC 2011 agents, in which 757 negotiation

ended with an agreement. For successful negotiations ending with an agreement, we separately analyze the negotiations per each opponent agent and demonstrate the average agent utility and opponent utility in Figure 3. It can be seen that *SAC Agent* managed to receive almost the same or better results against all ANAC opponents except *HardHeaded*. In addition, we are able to find agreements with the ANAC agents in approximately 60% of our negotiations, and in 65% of these agreements, we scored better than our opponents. It is worth noting that *HardHeaded* is a selfish and stubborn agent which does not change its offer until the very last moment. We would like to emphasize that it is hard to design a decent approach against such an unreasonable strategy. Besides, we can conclude that our agent achieved better results than the agents employing a more behavior-based approach. Especially against *the Negotiator*, *Gahboninho*, *BRAM* and *NiceTitfotTat*, our agent shows an outstanding performance.

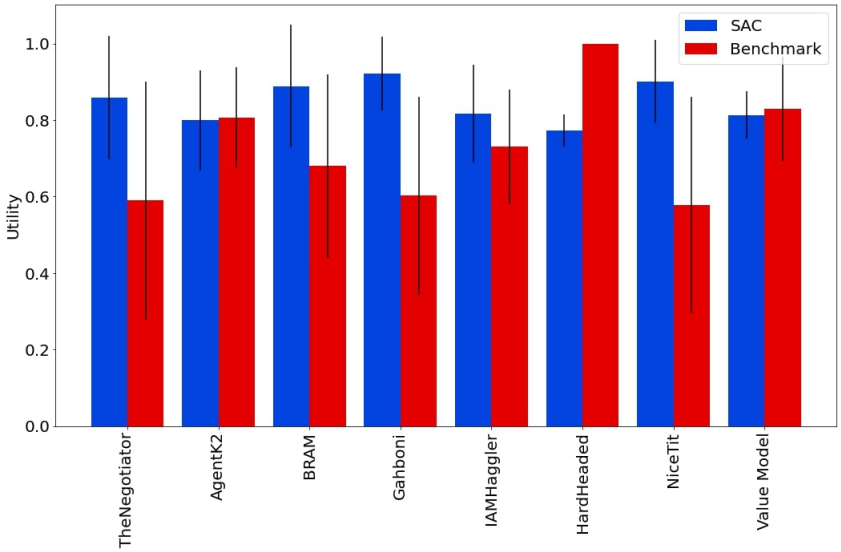

Figure 3. The average agreement results of our agent against ANAC 2011 agents over all test scenarios.

Furthermore, we investigate whether our agent learns to make compelling offers rather than random shooting and even negotiates better than another RL-based agent. Therefore, we pick the *Random Agent* and *RLBOA* as our baseline agents and compare their performance with ours. We adopt the same negotiation setting for these baseline agents. They negotiate with the ANAC 2011 finalist agents 10 times under the deadline of 100 rounds in the scenarios above. We categorize all negotiation scenarios in terms of the size of the domain and opposition of the scenario and report the average results with standard deviation in Table 5 and Table 6 elaborately. Note that best-performing values are boldface in the table. First, it can be observed that the winning rate of the *SAC Agent* is much higher than *RLBOA* and *Random Agent*, independent of domain size or scenario opposition, although their agreement rate is higher than that of the *SAC Agent*. When we group the negotiation scenarios in terms of domain size, the *SAC Agent* has a relatively lower agreement rate when the outcome space becomes larger compared to the small and medium-size domains (35% versus 75% and 84%). Independent of domain size average utility of the *SAC Agent* is about 85 out of 100, which is much higher than that of the *RLBOA* and *Random Agent* (approximately 85% versus 60%-73% and 27%-53%, respectively).

Table 5. Comparison of the *SAC Agent*, *RLBOA* and *Random Agent* in negotiations grouped by opposition size.

	Low opposition			**Medium opposition**			**High opposition**		
	SAC	**RLBOA**	**Random**	**SAC**	**RLBOA**	**Random**	**SAC**	**RLBOA**	**Random**
Avg. agreement utility	**83.49 ± 8.69**	78.57 ± 11.34	63.15 ± 17.6	**86.49 ± 11.93**	66.65 ± 13.27	53.79 ± 10.48	**86.68 ± 20.42**	40.7 ± 20.88	26.36 ± 13.0
Winning rate	**35.29**	14.08	4.38	**75.24**	13.46	0.42	**90.36**	8.12	0.0
Agreement rate	85.0	88.75	**100.0**	66.46	78.96	**100.0**	34.58	79.58	**100.0**
Acceptance count	272	284	**320**	319	379	**480**	166	382	**480**
Winning count	**96**	40	14	**240**	51	2	**150**	31	0
Total negotiation count	320	320	320	480	480	480	480	480	480

Table 6. Comparison of the *SAC Agent*, *RLBOA* and *Random Agent* in negotiations grouped by domain size.

	Small domains			**Medium domains**			**Large domains**		
	SAC	**RLBOA**	**Random**	**SAC**	**RLBOA**	**Random**	**SAC**	**RLBOA**	**Random**
Avg. agreement utility	**84.66 ± 12.71**	67.21 ± 14.92	53.18 ± 21.06	**86.79 ± 8.76**	73.37 ± 16.57	49.26 ± 12.08	**85.77 ± 20.42**	62.42 ± 20.88	27.75 ± 13.0
Winning rate	**56.77**	8.89	1.88	**65.47**	20.58	1.25	**89.38**	6.98	0.0
Agreement rate	65.78	84.38	**100.0**	69.69	75.94	**100.0**	35.31	80.62	**100.0**
Acceptance count	421	540	**640**	223	243	**320**	113	258	**320**
Winning count	**239**	48	12	**146**	50	4	**101**	18	0
Total negotiation count	640	640	640	320	320	320	320	320	320

When we group the scenarios in terms of their opposition (i.e. collaborative versus competitive), it can be seen that the *SAC Agent* and *RLBOA* have almost the same agreement rate in collaborative scenarios (i.e. scenarios with low opposition). However, the agreement rate of the *SAC Agent* decreases dramatically in competitive domains (i.e. scenarios with high opposition), but it achieves almost the same average utility that it can achieve in the cooperative scenarios around the utility of 85. On the other hand, the average utility of the *RLBOA* in competitive scenarios is much lower than that of the collaborative setting (40 versus 79). The average utilities of the agents against the ANAC 2011 agents are given in Figure 4. As expected, average utilities received by the agents drop when they negotiate with the same opponents in more competitive scenarios. It is worth noting that the performance of our agent does not change drastically and manages to find good agreements even in competitive scenarios. Note that the fourth graph shows the average utilities when we consider all negotiation domains. Except against the *HardHeaded*, our agent outperforms others with respect to the received utility in competitive scenarios while our agent performs better than other agents in collaborative domains with low opposition.

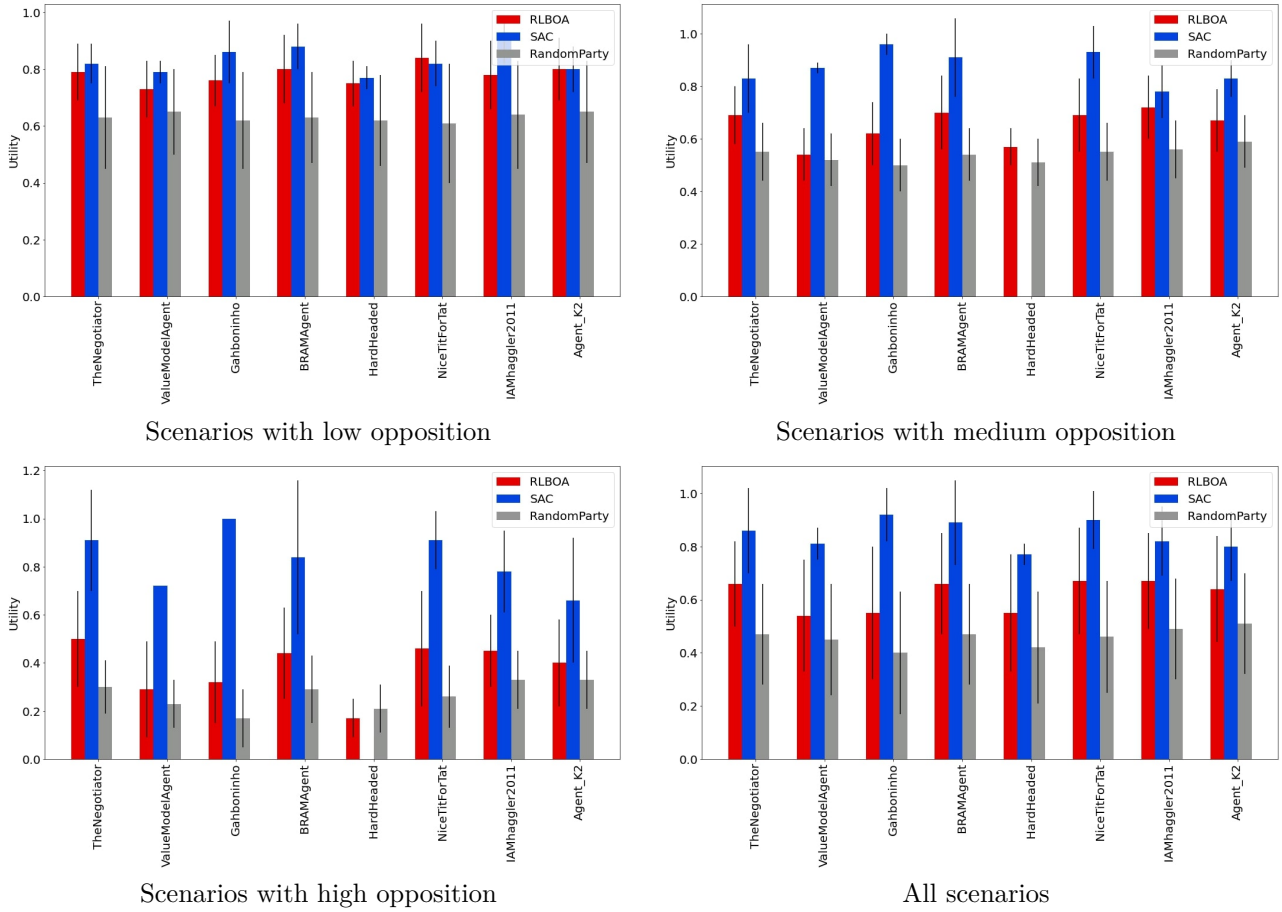

Figure 4. Average performance comparison of RL agent, RLBOA, and random agent against ANAC 2011 agents in scenarios with low, medium, and high opposition, respectively.

As far as all negotiation sessions are considered, *SAC Agent* find an agreement almost 60% of the negotiations. However, when it finds an agreement, it achieves a higher winning rate compared to RLBOA and

Random agents (approximately 64% versus 2% and 11%, respectively). That is, although *Random Agent* and *RLBOA* have higher agreement rates (100% and 81%, respectively), the average utility gained by them is much lower than that of the *SAC Agent* (approximately an average utility of 46 and 67 versus 85, respectively).

6. Conclusion & Discussion

This study presents an actor-critic RL-based bidding strategy for automated bilateral negotiation and shows empirically that actor-critic-style RL approaches could be successful in the negotiation domain using self-learning and behavior cloning. The performance of the proposed approach is evaluated two-fold. First, we tested how well the proposed RL strategy negotiated with the ANAC 2011 finalist agents and reported their pairwise performance results in terms of received utility. Second, we compare the performance of the *SAC Agent* with the baseline agents *RLBOA* and *Random agent* in terms of agreement rates, winning rates, and average received utility when they all negotiate with their opponents selected from the ANAC 2011 finalist agents. Our experimental results show that the proposed strategy can gain superiority over its opponents without requiring any opponent model or any other statistical knowledge. In addition, we analyze our agents in a variety of scenarios with varying domain sizes and opposition. Our agent generally performs well even when the problem becomes more complex (e.g., negotiating over a competitive negotiation scenario or a large-sized domain). The main strength of this work can be summarized as follows. First, we adopt a behavior cloning approach using the data generated from top bilateral negotiating agents in the ANAC. That helped reduce the training time of the SAC Agent and support its converge. In contrast to other work using Deep RL in the context of negotiation except [16], our state definition not only focuses on the current bid exchanges but also takes previous bid exchanges into account within a certain time interval. Consequently, the agent can comprehensively capture the behavior changes during the negotiation. Furthermore, our reward function does not consider only maximizing the individual utility but also minimizing the utility difference between negotiating parties. This may allow agents to increase their self-awareness regarding the negotiation outcome's fairness implicitly. Finally, we observed that the self-playing approach helps the agent explore its environment space more extensively than it does in other domains such as chess. Besides, our approach uses continuous action and state spaces, giving the agent a more fine-grained guideline while making its offer.

In future work, the behavior cloning structure of our agent can be further revised to make it superior to the rest of the agents. Moreover, while collecting behavior cloning data; we mostly considered the ANAC winner agents, which are highly competitive. In the future, we would like to categorize the opponents with respect to their collaborative behaviors and train a separate RL agent for each category. Here, another challenge would be to design an agent determining which RL agent is adopted against whom.

In addition, we are planning to test the performance of the developed RL-based negotiating agent against human negotiators. In a human-agent negotiation framework, we will ask human participants to negotiate with our agent and see whether our agent performs well even when it negotiates with a human participant instead of an automated agent. Moreover, the learned model could be fed with the human negotiation data in order to negotiate more human-like. Furthermore, it would be interesting to adapt the RL model for a multilateral negotiation setting where the agent has multiple opponents. In this case, we need to revise our state representation and reward function for capturing the entire negotiation dynamics.

Acknowledgments

This work has been supported by a grant from the Scientific and Research Council of Turkey (TÜBİTAK) with grant number 118E197. The contents of this article reflect the ideas and positions of the authors and do not necessarily reflect the ideas or positions of TÜBİTAK. We would like to thank Umut Çakan, Gevher Yesevi and Berk Buzcu for their technical support and also Prof. Erhan Öztop for his excellent guidance during this work.

References

[1] An B, Lesser V, Sim K. Strategic agents for multi-resource negotiation. Autonomous Agents and Multi-Agent Systems 2011; 23 (1):114–153. doi: 10.1007/s10458-010-9137-2

[2] Eran C, Keskin MO, Cantürk F, Aydoğan R. A decentralized token-based negotiation approach for multi-agent path finding. In: European Conference On Multi-agent Systems (EUMAS), Israel (Online); 2021. pp. 264–280.

[3] Sandholm T. Agents in electronic commerce: Component technologies for automated negotiation and coalition formation. Autonomous Agents and Multi-Agent Systems 2000; 3 (1): 73–96. doi:10.1023/A:1010038012192

[4] Botelho S, Alami R. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In: IEEE International Conference on Robotics and Automation; Detroit, MI, USA; 1999. pp. 1234–1239.

[5] Aydoğan R, Øzturk P, Razeghi Y. Negotiation for incentive driven privacy-preserving information sharing. In: PRIMA 2017: Principles and Practice of Multi-Agent Systems; Nice, France; 2017. pp. 486–494.

[6] Barslaag T, Gerding EH, Aydoğan 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, Singapore; 2015. pp. 190-197.

[7] Faratin P, Sierra C, Jennings NR. Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems 1998; 24 (3): 159–182. doi:10.1016/S0921-8890(98)00029-3

[8] Aydoğan R, Kafalı Ö, Arslan F, Jonker CM, Singh MP. Nova: Value-based Negotiation of Norms. ACM Transactions on Intelligent Systems and Technology 2021; 12 (4): 1–29. doi:10.1145/3465054

[9] Jonker CM, Aydoğan 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.

[10] Cao J. Research on electronic commerce automated negotiation in multi-agent system based on reinforcement learning. In: International Conference on Machine Learning and Cybernetics; San Antonio, Texas, USA ; 2009. pp. 1419–1423.

[11] Baarslag T, Hindriks K, Jonker C. A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations. In: Ito T, Zhang M, Robu V, Matsuo T (editors). Complex Automated Negotiations: Theories, Models, and Software Competitions, Berlin, Heidelberg: Springer, 2013, pp.229–233.

[12] Morii S, Ito T. AgentMR: Concession Strategy Based on Heuristic for Automated Negotiating Agents. In: Marsa-Maestre I, Lopez-Carmona M, Ito T, Zhang M, Bai Q, Fujita K (editors). Novel Insights in Agent-based Complex Automated Negotiation, Tokyo: Springer, pp. 181–185.

[13] Krimpen T, Looije D, Hajizadeh S. HardHeaded. In: Ito T, Zhang M, Robu V, Matsuo T (editors). Complex Automated Negotiations: Theories, Models, and Software Competitions, Berlin, Heidelberg: Springer, 2013, pp. 223–227.

[14] 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.

[15] Bagga P, Paoletti N, Alrayes B, Stathis K. A deep reinforcement learning approach to concurrent bilateral negotiation. In: The Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20); Yokohama, Japan (Online); 2020. pp 297–303.

[16] Sengupta A, Mohammad Y, Nakadai S. An autonomous negotiating agent framework with reinforcement learning based strategies and adaptive strategy switching mechanism. In: The 20th International Conference on Autonomous Agents and Multiagent Systems; London, UK; 2021. pp. 1163–1172.

[17] Nair A, McGrew B, Andrychowicz M, Zaremba W, Abbeel P. Overcoming exploration in reinforcement learning with demonstrations. In: IEEE 2018 International Conference on Robotics and Automation (ICRA); Brisbane, Australia; 2018. pp. 6292–6299.

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

[19] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge, Massachusetts: MIT press, 2018.

[20] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Thirty-fifth International Conference on Machine Learning; Stockholm, Sweden; 2018. pp.1861–1870.

[21] Lin L. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning 1992; 8 (3-4): 293–321. doi:10.1007/BF00992699

[22] Vecerík M, Hester T, Scholz J, Wang F, Pietquin O, et al. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. 2018, 1–10. arXiv:1707.08817v2

[23] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, et al. Playing atari with deep reinforcement learning. 2013, 1–9. arXiv:1312.5602.

[24] 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.

[25] 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.

[26] Fujita K, Ito T, Baarslag T, Hindriks K, Jonker C, et al. The Second Automated Negotiating Agents Competition (anac2011). In: Ito T, Zhang M, Robu V, Matsuo T (editors). Complex Automated Negotiations: Theories, Models, and Software Competitions, Berlin, Heidelberg: Springer, 2013, pp. 183–197.

[27] Williams CR ,Robu V, Gerding EH, Jennings NR. An overview of the results and insights from the third automated negotiating agents competition (ANAC2012). In: Marsa-Maestre I, Lopez-Carmona M, Ito T, Zhang M, Bai Q, Fujita K (editors). Novel Insights in Agent-based Complex Automated Negotiation, Tokyo: Springer, pp. 151–162.

[28] 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

[29] Tesauro G, Kephart J. Pricing in agent economies using multi-agent q-learning. Autonomous Agent and Multi-Agent Systems 2002; 5(1) :289–304. doi:10.1023/A:1015504423309.

[30] Sridharan M, Tesauro G. Multi-agent Q-learning and Regression Trees for Automated Pricing Decisions. In: Parsons S, Gmytrasiewicz P, Wooldridge M (editors). Game Theory and Decision Theory in Agent-Based Systems, Boston, USA: Springer, 2002, pp. 217–234.

[31] Oliveira F. Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning. International Journal of Management Science 2013; 47 (1): 116–126. doi: 10.1016/j.omega.2013.10.004

[32] 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.

[33] Lewis M, Yarats D, Dauphin YN, Parikh D, Batra D. Deal or no deal? End-to-end learning of negotiation dialogues. In: The 2017 Conference on Empirical Methods in Natural Language Processing; Copenhagen, Denmark; 2017. pp. 2443–2453.

[34] Kröhling D, Chiotti O, Martínez E. The importance of context-dependent learning in negotiation agents. Inteligencia Artificial 2019; 22 (63): 135–149. doi: 10.4114/intartif.vol22iss63pp135-149

[35] Razeghi Y, Yavuz O, Aydoğan R. Deep reinforcement learning for acceptance strategy in bilateral negotiations. Turkish Journal of Electrical Engineering and Computer Sciences 2020; 28 (1): 1824–1840. doi:10.3906/elk-1907-215

[36] Güneş TD, Arditi E, Aydoğan R. Collective Voice of Experts in Multilateral Negotiation. In: PRIMA 2017: Principles and Practice of Multi Agent Systems; Nice, France; 2017. pp. 450–458.

[37] Jin J, Song C, Li H, Gai K, Wang J et al. Real-time bidding with multi-agent reinforcement learning in display advertising. In: The 27th ACM International Conference on Information and Knowledge Management; Lingotto, Turin, Italy ; 2018. pp. 2193—2201.

[38] LaValle S, Branicky M, Lindemann S. On the relationship between classical grid search and probabilistic roadmaps. The International Journal of Robotics Research 2004; 23 (7): 673–692. doi:10.1177/0278364904045481

[39] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: The 27th International Conference on International Conference on Machine Learning; Haifa, Israel 2010. pp. 807–814.

[40] Paszke A, Gross S, Massa F, Lerer A, Bradbury J et al. Pytorch: An imperative style, high-performance deep learning library. In: Thirty-third Conference on Neural Information Processing Systems; Vancouver, Canada, 2019: 8024–8035.

[41] Aydoğan R, Yolum P. Ontology-based learning for negotiation. In: IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology; Milan, Italy; 2009: 177–184.

[42] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S et al. Soft actor-critic algorithms and applications. 2018, 1–17. arXiv:1812.05905.

[43] Tunalı O, Aydoğan 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.