AN ACTOR-CRITIC REINFORCEMENT LEARNING APPROACH FOR BILATERAL NEGOTIATION

A Thesis Proposal

by

Furkan Arslan

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Computer Engineering

in the
Department of Computer Engineering

Özyeğin University
October 2021

AN ACTOR-CRITIC REINFORCEMENT LEARNING APPROACH FOR BILATERAL NEGOTIATION

Approved by:

Assistant Professor Reyhan Aydoğan,
Advisor
Department of Computer Engineering
Özyeğin University

Professor Erhan Öztop
Department of Computer Engineering
Ozyegin University

Associate Professor Emre Uğur
Department of Computer Engineering
Boğaziçi University

Date Approved: 27 August 2021

To my lovely family

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 thesis 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 determining 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 not only take 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.

ÖZETÇE

Bir çıkar çatışması olduğunda yazılım aracılarının karşılıklı bir anlaşma bulmak amacı- yla birbirleriyle yaptığı pazarlık için etkili ve akıllı bir teklif stratejisi tasarlamak otomatik müzakeredeki en zorlayıcı araştırma zorluklarından biridir. Bu tez de, elle hazırlanmış bir karar verme modülü tasarlamak yerine, ikili bir müzakerede hangi teklifi sunulacağını öğrenen bir aktör-eleştirmen takviyeli öğrenme yaklaşımını benimseyen yeni bir teklif stratejisi önerilmiştir. Teklif verme yaklaşımı için Soft Actor-Critic (SAC) adı verilen bir entropi pekiştirmeli öğrenme yaklaşımı kullanılmıştır. Önceki teklif değişimlerine ve kalan süreye dayalı olarak gelecek teklifin hedef değerini belirleyen SAC modeli eğitmek için bir kendi kendine oynama yaklaşımı kullanılır. Ayrıca, öğrenme sürecini hızlandırmak için davranış klonlama adı verilen bir taklit öğrenme yaklaşımı benimsenmiştir. Bunlara ek olarak, müzakerenin sonunda yalnızca temsilcinin aldığı teklif değerini değil, aynı zamanda rakibin teklif değerinide kullanan yeni bir ödül işlevi tanıtıldı.

Bu tez kapsamında geliştirilen SAC ajanı ampirik olarak değerlendirildi. Bu amaçla, büyüklük ve zorluk bakımından farklı alanlarda seçilen çeşitli muhaliflere karşı çok sayıda pazarlık oturumları yürütüldü. Ajanın performansı rakipleriyle ve aynı rakiplerle pazarlık eden temel ajanların performansıyla karşılaştırıldı. Ampirik sonuçlar, temsilcimizin, rakip veya alan hakkında önceden herhangi bir bilgi gerektirmeden farklı pazarlık senaryolarında zorlu rakiplere karşı başarılı bir şekilde pazarlık ettiğini göstermektedir. Ayrıca, başarılı pazarlık sonunda alınan teklif değeri göz önünde bulundurulduğunda temel aracılardan daha iyi sonuçlar elde etmektedir.

ACKNOWLEDGEMENTS

"We had to start somewhere, either succeed or fail, and then build what we knew as we went along."

– Homer Hickam

First of all, I would like to thank Assistant Professor Reyhan Aydoğan for her limitless support, time, guidance, and motivation whenever I need. I would also like to thank all members of Interactive Intelligent Systems especially Umut Çakan and Berk Buzcu. I am deeply grateful to Professor Erhan Öztop and Associate Professor Emre Uğur for participating in my thesis jury and giving me feedback. Last but not least, I would love to thank to my dearest family. It would be impossible to complete without their unconditional support over those years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

γ	Discount factor of future reward
$\mathcal{B}$	An offer
$\mathcal{I}$	List of issues
$\mathcal{O}$	All possible outcomes
$\mathcal{O}^*$	the set of all possible offers in the negotiation domain
k^a	The first bid of an agent a which is mostly a $\mathcal{B}$ that has maximum utility
π	A function mapping states to actions
A	Action
a_t	Current Action
E	Environment
e	Epsilon value that controls concession rate of time-dependent agents
G	Expected return of an episode
i	Issue
M	The maximum amount by which an agent can change its imitative behaviour
$P(s, s^{'})$	Transaction Probability
PE	Policy Evaluation
PI	Policy Improvement
r_t	Immediate Reward
S	State

s_t,s	Current State
$s_{t+1}, s^{'}$	Next State
T	Terminal State
t	Current time of the negotiation session
$V_i(.)$	The valuation function for issue i
w_i	The importance of the negotiation issue i for the agent

LIST OF ACRONYMS/ABBREVIATIONS

AN Automated Negotiation.

ANAC the Automated Negotiating Agents Competition.

BC Behavior Cloning.

DDPG Deep Deterministic Policy Gradient.

DRL Deep Reinforcement Learning.

GENIUS General Environment for Negotiation with Intelligent Multi Purpose Usage Simulation.

MADDPG Multi-agent Deep Deterministic Policy Gradient.

MDP Markov Decision Process.

ReLU Rectified Linear Unit activation function.

RL Reinforcement Learning.

RLBOA A Modular RL Framework for Autonomous Negotiating Agents.

SAC Soft Actor Critic.

SOAP Stacked Alternating Offers Protocol.

CHAPTER I

INTRODUCTION

Automated agreement technologies provide promising solutions to a variety of problems such as resource allocation [1], path planning [2], and resolving conflicts [3], and have a vast range of applications including electronic commerce [4], coordination in robotics [5], privacy management for information sharing systems [6], and so on. Thus, there is increasing attention to the field of automated negotiation in multi-agent systems. Researchers have designed various negotiation strategies [7, 8, 9] and protocols [10] 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, a variety of bidding strategies have been proposed so far in the literature. While some researchers focus on designing agents that are capable of negotiating with others only on a particular domain[11], 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.

Accordingly, some behavior-based bidding strategies have been designed to reciprocate the opponent's behavior during the negotiation [12]. This process requires learning the opponent's preferences. Consequently, a vast number of studies have focused on opponent modeling using some heuristic approaches [13, 14] and adopting some machine learning techniques such as Bayesian learning and some concept-based learning methods [15, 12]. Apart from this, 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.

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 [16]. 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 make 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 moves of the agent. A more sophisticated deep RL approach could be developed by considering continuous state and action spaces. Most recent works [17, 18] 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 to the best of our knowledge. 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 (BC) [19] approach.

Accordingly, this article proposes a novel bidding strategy using an actor-critic approach named Soft Actor Critic (SAC). The proposed strategy called *SAC Agent* uses the power of maximum entropy reinforcement learning to learn how to negotiate independent of scenario and opponent from scratch. We intend to find the most appropriate target utility to make offers without requiring any opponent modeling 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. To sum up, 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 thesis is organized as follows: Section 2 briefly discusses the related works. Section 3 provides the necessary background for negotiation and reinforcement learning 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 the explanation about the methodologies used for the generalization of the model. Finally, Section 6 concludes the thesis and discusses the future work.

CHAPTER II

RELATED WORK

Deep Reinforcement Learning (DRL) has recently led to a wide range of successes in learning policies for sequential decision-making problems in simulated environments such as playing Atari games[20] and defeating the best human player at the game of Go[21], as well as robotic tasks such as helicopter control[22], screwing a cap onto a bottle[23], or door opening [24] and so on.

Along with these successes of RL, it has also started to be used in the field of Automated Negotiation (AN) in recent years. [25, 26] have shown the success of the Q-learning algorithm, which is the most widely used RL algorithm in price negotiation. Tesauro and Kephard explore how AN agents can use RL algorithms such as Q-learning to make economic decisions such as setting prices in a competitive market [25]. In addition, Tesauro and Kephard investigate whether RL algorithms such as Q-learning that consider long-term gains can find the right optimal in the economic model. In other words, an agent must be able to predict the long-term consequences of his own actions, the actions of other agents and model other agents with equivalent ability accurately. For this purpose, they make two competing vendors use Q-learning simultaneously in three moderately realistic economic models (i.e., two sellers and one buyer setting). Since the state space in these three economic models is not very large, it is sufficient to use lookup tables to represent the Q functions. Compared to direct modeling tool policies, the Q-learning approach is more extensible to the situation of vast economies with many competing vendors. In the many competing vendor cases, each agent needs a single Q function using a nonlinear function estimator such as neural networks, whereas, in the policy modeling approach, each agent must maintain a policy model for every other agent. Their experimental results show that Q-learning

policies lead to more income and make agents behave more rationally than simple policymakers.

Moreover, Sridharan and Tesauro argue that it is difficult for automated agents to perform highly well with a hard-coded behavior approach against agents implementing different strategies in complex, ever-changing, multi-agent environments [26]. Therefore, they propose an approach based on Q-learning with a regression tree approach that can learn over time and adapt to different negotiation environments for single-agent and multi-agents. With this approach, it is aimed that automatic agents learn pricing strategies in a competitive market. In this market environment, two different sellers make price offers, and the buyer chooses the most suitable offer among these two offers. During a negotiation, sellers update their bids in turn according to their competitor's bid until they determine their final bid. Then the buyer chooses the offer that is suitable for itself. The value of this offer is used as a reward for the Q-learning model. Regression trees, which are used as an alternative function approximation to neural network models, are intended to shorten the training time compared to neural networks and produce policies of at least equal quality to those produced by a neural network. Experimental results support that the proposed approach is the first successful combination of regression trees and Q-learning combinations. In addition, the advantages of regression trees over the neural network approach are clearly demonstrated. As a result, Q-learning approaches are shown to be a promising approach in multi-agent price economy approaches.

RL also shows great success on agenda-based negotiation systems. Papangelis and Georgila present a multi-issue negotiation dialogue model that includes a reinforcement learning policy to work in a wide variety of negotiation environments and perform well against negotiation partners whose behavior has not been observed before [27]. The RL model is built on deep Q-learning architecture. The RL model is trained and evaluated against a handcrafted agenda-based policy agent in different negotiation environments. It is used for the first time in agenda-based

policy negotiation scenarios. The negotiation profile models contain information about persuadability, available arguments, and preferred/acceptable values (possible outcomes) for each issue. The learned RL model ultimately outperformed the agenda-based model.

Furthermore, Zou, Zhan, and Shao show that long-term planning agents are more successful than short-term planning agents in negotiation by using Q-learning algorithm [28]. The negotiating agents which use myopic approaches to determine their actions by enormously depending on the details of the negotiation process and the opponent's characteristic. It is challenging to come up with a proper negotiation model with these myopic approaches. In addition, a long-term negotiation approach is required to develop a negotiation model against agents who can use different negotiation behavior models in a negotiation process. Reinforcement learning is a suitable method for determining the agent's optimal strategy for scenarios where competition and domain knowledge are limited, such as the bargaining domains. Zou, Zhan, and Shao present a new negotiation approach that adopts reinforcement learning to calculate each strategy's reward and then uses replicator dynamics to adjust the probability of strategies. The learning model is based on the alternate offering mechanism in which agents adopt three different time-dependent concession strategies (frugal strategy, cool-headed strategy, anxious strategy) for both seller and buyer. Another important aspect of this approach is applying a new reward model that uses both the present and weighted historical rewards. Their results show that this combined model achieves higher rewards, shorter negotiation time, and a lower degree of the greediness of strategies than the classic evolution model.

RL has shown promising results not only in issue-based negotiations but also dialog-based negotiations [29]. If this negotiation occurs in natural language, agents must develop a bargaining strategy and properly articulate it in natural language. In this, bargaining needs to be formulated and then realized verbally.

To this end, Lewis *et al.* collects the first large dataset of natural language negotiations between two people [29]. This dataset contains 5808 human conversations negotiating with each other. They train a recurrent neural network that aims to mimic human behavior by using this dataset. While the trained model successfully produces natural language bargaining dialogues, they failed to develop a proper negotiation strategy. In order to give these agents the ability to bargain, they developed two different methods: self-play RL and supervised updates. They first demonstrate that agents significantly improved their ability by using self-play. A supervised update performs after reinforcement learning updates to ensure that pre-trained models use natural language when the agents are negotiating with each other. In this way, it has been shown for the first time that an agent who uses reinforcement learning and negotiates with dialogue is more successful than supervised learning models. The second proposed improvement is a new form of dialog planning called dialogue rollouts. Thanks to this new form, the reward of dialogues produced by the agent can be estimated. By using reward maximization, the agent's performance significantly improves against both human opponents and other agents.

Apart from those approaches, Schmid *et al.* and Jin *et al.* incorporate multi-agent RL approaches into negotiation [30]. Real-time advertising allows advertisers to bid based on user's interests. Jin, Song and Li *et. al.* claim that the relationship between the advertisement and the user's interests must be recognized correctly to maximize the revenue of an advertisement. Consequently, they propose a bid optimization approach based on multi-agent reinforcement learning. They cluster bidders and consumers. The bidding agents can work at the cluster level by assigning a strategic bid agent to these clusters. The proposed multi-agent formulation is based on interactions between the consumer cluster and cluster-level bidding agents, as well as the interactions among bidding agents. This multi-agent approach allows agents to work together while competing with each other. They proposed the Distributed Coordinated Multi-Agent Bidding (referred to as

DCMAB) based on the Multi-agent Deep Deterministic Policy Gradient (MADDPG) technique. The method stabilizes convergence by feeding all agents' bid actions to the action-value (Q function). Here, the transaction value of each bidding agent evaluates future action-value based on the actions of all agents rather than just its action. An empirical work with real-world data has demonstrated the effectiveness of their approach. The results show that their approach significantly outperform the single-agent and bandit approaches of cluster-based bidding. In addition, coordinated bidding achieves better overall goals than bidding agents who are purely self-interested.

Moreover, some researchers focus on learning the content of the offer rather than the target utility of the coming offer. For instance, Sunder *et. al.* analyze the behavior of RL negotiation agents that have prosocial and selfish behavior trained by playing each other [16]. In their work, domain of each negotiation issue is binary and their RL approach aims to learn their assignments while constructing the current offer. They adapt the reward functions in a way that the learned RL model acts as either prosocial or selfish agent. To do this, agents negotiate themselves with each other. Finally, a meta agent with a mixture of behaviors is constructed. The results show that meta agents are capable of emulating human behavior successfully.

Kröhling, Chiotti, and Martinez claim that agent can benefit from using the negotiation context and important external variables, especially when modeling the opponent's strategy, behaviors, and assumptions [31]. They propose a new negotiation 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. The agent also internally models contextual variables and learns to leverage this knowledge by playing against himself with RL. Strawberry shows quite competitive results in a heterogeneous environment composed of several agents, even though it does not explicitly model its opponents.

Another good example of RL in automated negotiation is that Bakker *et. al.* use tabular Q-learning to learn when and to what extent it should concede during the negotiation [32]. They define ten utility bins (e.g., [1–0.9), [0.9–0.8), and so on). Their agent will learn whether to stay in the current bin or move up or down to neighbor bin while determining the target utility of the current offer. This approach adopts discrete state and action space to reduce 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. Moreover, Razeghi *et. al.* propose to use a Deep RL approach to learn when to accept the opponent's offer in automated bilateral negotiation [33]. Their experimental results show that the proposed RL-based acceptance strategy performs as well as the AC-next strategy, the most widely-used strategy. Bagga,Paoletti and Alrayes *et. al.* introduce an RL approach for both acceptance and bidding strategies. In this work, a model-free actor-critic(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 [17, 34]. 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. For example, if the Q agent converses 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 proposes using an actor-critic approach for bidding and a mechanism for selecting the most appropriate trained RL agents among the multiple ones [18]. They train the proposed RL agents by making them negotiate with a fixed set of opponents such as Boulware, nice TFT, conceder in several negotiation scenarios. Similar to our approach, they also use a SAC bidding strategy

with a fixed acceptance strategy. That work advocates to train different RL agents by training with different opponents and select 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 approach as proposed in [35].

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 (BC) 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. Table 1 present a comparison of some selected works.

Table 1: Functionality comparison of the existing work.

	Zou *et. al.* [28]	Lewis *et. al.*[29]	Jin *et. al.*[30]	Sunder *et. al.* [16]	Bakker *et. al.* [32]	D. Kröhling *et. al.* [31]	Razeghi *et. al.* [33]	Bagga *et. al.* [17]	Sengupta *et. al.* [18]	SAC Agent
Domain independent	√	√	×	×	√	×	√	√	√	√
Acceptance Criteria	ACnext	×	ACnext	ACnext	ACnext	ACnext	RL Agent	RL Agent	ACcombi	ACnext
RL Algorithm	Tabular Q-learning	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	×	×	×	×	×	×	√	√	×	×
Multi Agent	×	×	√	√	×	×	×	×	√	×
Meta Agent	×	×	×	√	×	×	×	×	√	×

CHAPTER III

BACKGROUND

In this section, we provide the necessary background on automated negotiation (Section 3.1) and reinforcement learning (Section 3.2).

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 $\mathcal{B}$, 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, $\mathcal{B}_i$ represents the value for issue i in offer $\mathcal{B}$, 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.

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

3.1.1 Negotiation Protocol

The interaction among agents during an automated negotiation is governed by the Stacked Alternating Offers Protocol (SOAP) [36], which extends alternating offers protocol [37] to more than two negotiating parties. In SOAP, 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($V_i(.)$), 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.

3.1.2 Negotiation Strategies

In each negotiation round, an agent generates its current offer using a set of functions to determine the agent's bidding behavior by considering some criteria (e.g., time, outcome, etc.). The agents follow a bidding strategy by taking into account their own preferences as well other factors such as the opponent's behavior, estimated preferences, and time. Faratin *et al.* categorize them into families as follows [8]: time-dependent negotiation strategies (Section 3.1.2.1) and behaviour dependent negotiation strategies (Section 3.1.2.2).

3.1.2.1 Time Dependent Negotiation Strategies

In automated negotiation, parties need to come up with an agreement within a certain time, *deadline*. Time-based negotiation strategies assume that the agent will concede over time [8]. While some agents concede fast during the negotiation, others have a tendency to concede only towards the end of the negotiation. The speed of concession and to what extent the agent will concede will be determined by a negotiation tactic. Accordingly, Faratin *et al.* define two tactics: *Conceder* and *Boulware*. Both tactics rely on a target utility function defined by Equation 2 in polynomial form with an epsilon(e) value. This epsilon value determines the rate of concession of the agent. The more the value of e becomes bigger, the more agent concedes in time, while if the value of e is less than 1, the agent will prefer a tougher bidding approach. If the value of e is 1, the agent linearly concedes until the deadline is over. The k^a in the Equation 2 is a constant denoting the utility

of the first offer (i.e., mostly corresponds to the maximum utility for the agent in the outcome space), which is between zero and one.

$$\mathcal{U}^a(target) = \mathrm{k}^a + (1 - \mathrm{k}^a) * \mathrm{t}^{1/e} \tag{2}$$

There can be an infinite number of agent tactics employing different epsilon values in the time-dependent negotiation family; however, two sets of agents are well-known time-dependent agents in negotiation literature [8]. The *Boulware agents* use $e < 1$, which means that the agents do not concede until a certain time; afterward, they concede fast up to their reservation value. On the other hand, *Conceder agents* use $e > 1$, which means that agents constantly concede over time.

3.1.2.2 Behaviour Dependent Negotiation Strategies

Agents may consider their opponent's moves while determining their target utility. The agents mimic their opponent's move relying on their opponent modeling when they employ a behavior-dependent strategy [8]. The strategies in this family may differ with respect to what extent they mimic their opponent. In this thesis, the target utility estimation is done by following Equation 3 where δ denotes the number of steps we will consider in the bid history, $U_{b \to a}$ denotes utility of the opponent's offer for the agent, and $U_{a \to b}$ corresponds to the utility of agent's previous offer. Here, U^a_{min} and U^a_{max} represents the minimum and maximum utility for our agent in the given negotiation domain, respectively. Moreover, the value of the parameter of s changes the behaviour by increasing or decreasing the target utility by a random amount, where R(M) is a function that generates a random integer in the interval [0, M]. Here, M is the maximum amount by which an agent can change its imitative behaviour. The condition of applicability is $n \geq 2\delta$.

$$\mathcal{U}_{a\to b}^{t_{n+1}}(target) = min\left(max(\mathcal{U}_{a\to b}^{t_{n-1}} + (\mathcal{U}_{b\to a}^{t_{n-2\delta}} - \mathcal{U}_{b\to a}^{t_{n-2\delta+2}}) + (-1)^s R(M), \mathcal{U}_{min}^{a}), \mathcal{U}_{max}^{a}\right) \tag{3}$$

$$s = \begin{cases} 0 & \text{if } \mathcal{U}^a \text{ is decreasing} \\ 1 & \text{if } \mathcal{U}^a \text{ is increasing} \end{cases} \tag{4}$$

3.2 Reinforcement Learning (RL)

Reinforcement Learning is a goal-oriented optimization technique that aims to learn a *policy* by interacting with the environment through trial and error [38]. A policy, which is a function mapping states to actions $\pi : S \to A$, specifies what actions to be taken in the underlying states to maximize future expected rewards. We consider the Markov Decision Process 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 memory 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 restart after the *terminal state* T or *final step*. That is, a new subsequence starts in each episode. Note that an episode corresponds to the fundamental steps of a game from the beginning to the ending. 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 5 where γ is a discount factor for future rewards, which is between 0 and 1. In the context of the negotiation, that corresponds to maximizing the future utility

of the negotiation outcomes.

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + r_{t+4} + \cdots + r_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{5}$$

RL uses two important function concepts derived from the Bellman equations. The first is the value functions 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 6 [38] 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 is any time step. The second concept, the action-value function called Q function is similarly defined in Equation 7.

$$v_\pi(s) = E_\pi[G_t|\mathcal{S}_t = s] = E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k|\mathcal{S}_t = s\right], \text{for all s } \in \text{S} \tag{6}$$

$$q_\pi(s,a) = E_\pi[G_t|\mathcal{S}_t = s, \mathcal{A}_t = a] = E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k|\mathcal{S}_t = s, \mathcal{A}_t = a\right] \tag{7}$$

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* as shown Equation 8 and *policy improvement* as shown Equation 10 [38], which is based on the greedy policy definition given in Equation 9. While the former is about computing the value function for a policy, the latter uses the value function to find a better policy [39]. This iteration can be demonstrated like in Equation 11 where PE and PI denotes a policy evaluation, a policy improvement respectively.

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right] \tag{8}$$

$$\pi'(s) = \arg\max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_{\pi'}(s')\right] \tag{9}$$

$$v_{\pi'}(s) = \max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_{\pi'}(s')\right] \tag{10}$$

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} \cdots \xrightarrow{PI} \pi_* \xrightarrow{PE} v_* \tag{11}$$

Lastly, the RL approach is different from the supervised learning and unsupervised learning approaches of machine learning. The most distinctive feature distinguishes reinforcement learning from other machine learning models is that it uses educational information that evaluates the action steps taken with the most appropriate actions' instructions. It requires active research, namely the search for good behavior explicitly. In supervised learning, we create labeled training and tests set where the training set is used to fit a model capturing the structure of the training instances, and the test set is used for evaluating the performance of that model. In this way, new unlabeled samples can be predicted with the help of the model. In RL, the reward is the only supervision available.

CHAPTER IV

METHODOLOGY

This master thesis aims to design and develop a smart autonomous agent, who has no prior knowledge of the field, learns how to bargain by self-play. This learning technique will be carried out with an actor-critic method called Soft Actor-Critic (SAC) [39], which is one of the state of the art reinforcement learning techniques currently available. The main intuition is to make our agent negotiate with itself repeatedly to explore finite continuous states over time to learn 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. In the following sections, we describe our bid generation approach (Section 4.1), we present how we represent state, action, and rewards for the bilateral negotiations (Section 4.2) and explain how we used the behavior cloning approach (Section 4.3).

4.1 Bid Generation by Using SAC

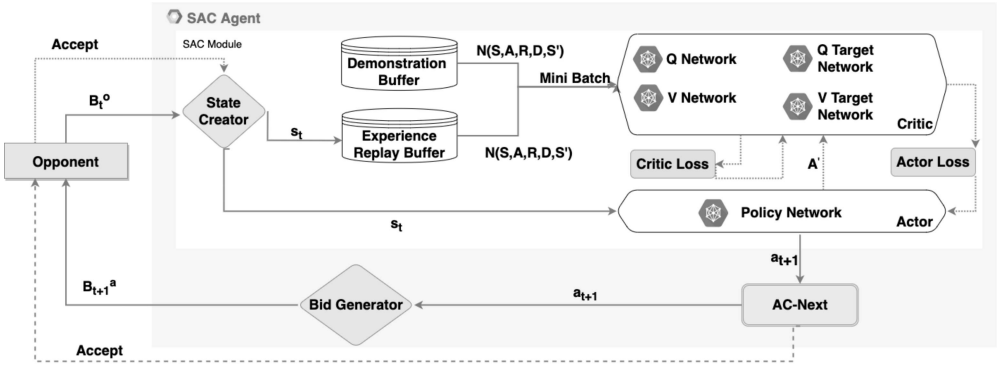


Figure 1: Overall structure of the proposed SAC agent

SAC is an off-policy actor-critic algorithm based on maximum entropy reinforcement learning (RL). The main strength of SAC is that the actor maximizes both expected return and entropy simultaneously that helps to find the balance between exploration and exploitation, where most RL methodologies suffer from. 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 to learn 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* [40] is used. An experience replay buffer can be considered as accumulated negotiation traces from the previous and current negotiations. In the proposed approach, the *SAC agent* uses both demonstration and experience replay buffers. The SAC model takes a random mini batch from these buffers to update its policy at each time step.

SAC combines off-policy actor-critic training with a scholastic actor and adopts an entropy maximization objective[41]. 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. As defined in [39], Equation 12 shows the policy objective with entropy maximization where $\mathcal{H}$ denotes to entropy function and α determines relative importance of the entropy.

$$\pi* = \arg\max_{\pi} \sum_{t} E_{S_t, a_t \sim p_\pi}[r(S_t, a_t) + \alpha \mathcal{H}(\pi(.|S_t))] \tag{12}$$

Recall that, actor critic approaches perform concurrent updates of the value function and policy function. The policy is considered as the *Actor* and the value function as the *Critic* in actor-critic RL approaches. The *Critic* element of the SAC algorithm consisting of Q and V Networks adopts the clipped double Q network approach [42, 43] 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 consists of a Gaussian policy network learning the optimal stochastic policy to determine the best action to take at the underlying state.

As we described before, state information is given as an input into the SAC model and the model itself produces the target utility as utility of our agent's next 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* [44]. Otherwise, it will generate an offer with the target utility by using Genius bid generation algorithm which take an target utility as input and produces corresponding bid considering agent's preferences. If there is no available offer with that utility in the given scenario, the bid generation algorithm chooses an offer whose utility is the closest to the estimated target utility.

4.2 State, Action And Reward Definitions

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 [45], 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 13 where U_i^a and U_i^o denote the utility of the agent's offer and the opponent's offer at time step i respectively, and $\mu \in [0.1 - 0.9]$ denotes the normalized negotiation time.

$$S_t = (\mu, \cup_{i=t-1}^{t-4}(U_i^a, U_i^o)) \tag{13}$$

As the reward signal of the RL agent, mostly the received utility of the agreement by the agent is considered [32, 17]. 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 4.2 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.

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

4.3 Behaviour Cloning(BC)

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 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 algorithm tends to be more stable than other off-policy approaches, independent of hyper-parameters. 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) [46] to reduce the training time of the SAC agent and support its converge. The demonstration replay buffers store same format as experience buffer with $< s_t, a_t, r_t, s_{t+1}, d_t >$. According to this methodology, we use both demonstration and replay data while performing actor and critic updates for each mini-batch.

We use the same behavior cloning approach with Vecerík *et. al.* [47]. Equation 14 shows the behaviour cloning loss (Equation 14) computed only on the demonstration examples for training actor, which introduced by Ashvin Nair *et. al.* [19]. This loss aims to improve the learned policy without going beyond the demonstration data.

$$L_{BC} = \sum_{i=1}^{N_D} ||\pi(S_i|\theta_\pi) - a_i||^2 \tag{14}$$

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}.

CHAPTER V

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 GENIUS [48]. 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 the the Automated Negotiating Agents Competition (ANAC), which is a yearly organized international competition where participants design negotiating agents in the context of a given research challenge [46]. Since the research focus is designing effective negotiation strategy, particularly for bilateral negotiations, we only consider the successful agents of the ANAC 2011-2012 [49, 50] in our evaluation.

In the following sections, we describe the training session of the *SAC Agent* (Section 5.1), study the effect of behavior cloning and reward function (Section 5.2), and present the experiment results in a test setting where *SAC Agent*, *Random Party* and *RLBOA Agent* negotiate separately with the opponents consisting of the ANAC 2011–2012 finalist agents (Section 5.3).

5.1 Evaluation Setup

In this thesis, we created two evaluation settings to train and test our *SAC Agent*. In the first setting, we train our agent by making it negotiate with ANAC 2011 finalist agents and test its performance against ANAC 2012 finalist agents. In our second setting, ANAC 2012 finalists are used to train our agent while ANAC 2011 finalists are used to evaluate its performance. In the following sections, we explain how we train SAC model for each setting (Section 5.1.1) and give the details of

the negotiation tournament setting in order to evaluate the performance of *SAC Agent* (Section 5.1.2).

5.1.1 Training Setting

To train our RL agent, we first constitute a demonstration buffer by using negotiation transactions obtained from the negotiations among the ANAC 2011 and 2012 winning agents top agents to adopt the imitation learning approach. We generate demonstration buffer 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 foods, drinks, location, invitations, music, and clean up respectively), which result in $4^5 * 3 = 3072$ possible outcomes. In total, 40146 and 40618 negotiation transactions have been collected from the ANAC 2011 winning agents and the ANAC 2012 winning agents, respectively. Note that a negotiation transaction corresponds to bid 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 itself on the *party domain* in total 54000 negotiation sessions.

In the first experimental setup, only negotiation transactions of ANAC 2011 [49] top agents are used for constructing the demonstration buffer. We train our agent with this buffer along with its experience replay buffer. Particularly, this buffer includes bilateral negotiation transactions among *The Negotiator*, *Gahboninho*, *Hardheaded*, *BramAgent*, and *IAMHaggler2011* from total 508 different negotiation sessions on the *party* domain. Note that each agent negotiates with themselves as well as other agents in a bilateral fashion by adopting a different preference profile. The deadline for each negotiation session is set to 100. It means that each agent needs to come up with an agreement in 100 rounds.

In the second setup, the demonstration buffer is created from the negotiation transactions produced by successful ANAC 2012 [50] top agents to adopt the imitation learning approach. To achieve it, we ran in total 512 different negotiation

sessions among CHUKAgent [51], AgentLG [50], OMACAgent [52], TheNegotiatorReloaded [53], and AgentMR [13] on the *party domain*. 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 in negotiation transactions.

In our SAC implementation, each neural network consists of three fully connected layers. The TanH activation function is used for the output layer while the Rectified Linear Unit activation function [54] is employed for the other layers. The Grid search [55] 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 hyper-parameters for the SAC implementations are given in Table 2. Note that the most of the hyper-parameters are the same as the original implementation of the SAC. 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. For agent implementation, we havely use Medipixel RL algorithms[56] that an one of the great Pytorch [57] library that includes RL model implementations.

Table 2: The hyper-parameters of the SAC implementation.

Parameters	Value
Gamma	0.99
Policy Update Interval	2
Tau	5.10^{-3}
Experience Replay Buffer	1.10^{6}
Initial Random Action	1.10^{5}
Actor Learning Rate	3.10^{-4}
Value Learning Rate	3.10^{-4}
Q Learning Rate	3.10^{-4}
Entropy Learning Rate	3.10^{-4}
Weight Entropy	1.10^{-3}
Weight Decay	0.0
Batch Size	256
Demo Batch Size	128
Lambda1(λ_1)	1.10^{-3}
Lambda2(λ_2)	1/128

For the both settings, these demonstration buffers and experience buffer are used to train the *SAC Agent* by making it negotiate itself on the *party domain* in total 54000 negotiation sessions.

5.1.2 Negotiation Tournament Settings

In order to demonstrate the generality of our approach, we tested our agents by running tournaments against the ANAC 2011-2012 finalist agents, Random Party, and an RL-based negotiating agent called RLBOA [32] in a variety of scenarios varying size 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 3. The opposition corresponds to the distance from the Kalai-Smorodinsky point to the point of maximum utility for both parties as defined in [58]. A high opposition value means 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 evaluation setting.

Table 3: 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 and negotiates with their opponent bilaterally 10 times over each negotiation scenario. The deadline for each negotiation session 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.

As baseline agents, we consider the *Random Party* to show that our agent can learn how to offer and consequently is able to 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. A brief description of those agents are given below:

- ***Random Party***: It is a baseline agent that generates offers randomly at each negotiation step.
- ***RLBOA*** [32]: RLBOA is a modular framework that facilitates the creation of autonomous negotiating agents using RL. This framework implements an agent using 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. We ran our tests using the trained models in the RLBOA repository.

We tested the success of our approach by performing tournaments in the Genius environment with our agent in the first setting against ANAC 2012 [50] agents, *RLBOA Agent*, and a *Random Party*. The brief descriptions of ANAC 2012 finalist agents are given below:

- **CUHKAgent**: The agent uses an adaptive negotiating strategy to predict the opponent's strategy and preference at a high level and make an informed decision accordingly [51].
- **AgentMR**: Agent MR uses a heuristic strategy bidding approach to find proper bid with high utility at an early stage and it controls its concession based on the opponent's behavior [13].

- **OMACAgent**: OMAC agent adopts opponent modeling and adaptive concession strategy. It uses two mathematical technique of wavelet decomposition and cubic smoothing spline for predicting opponent modeling. Concession's rate of the agent is arranged on the basis of the expected utilities of forthcoming counter-offers [52].

- **TheNegotiationReloaded**: TheNegotiationReloaded uses a heuristic approach by taking the opponent's strategy and domain characteristics into account to optimize its negotiation strategy [53].

- **AgentLG**: This agent took the second place in discounted domains of the final round of the Automated Negotiating Agents Competition 2012 [50].

- **BramAgent 2**: The agent aims to reach an agreement by collaborating with the opponent. As time goes on, it searches for a win-win solution by conceding. This model is the improved version of the BramAgent in the ANAC 2011 [50].

- **IAMhaggler2012**: This agent uses a Gaussian process regression approach to estimate the concession moves of its opponents. Accordingly, it generates its offers. This model is the improved version of the IAMhaggler in the ANAC 2011[50].

- **MetaAgent 2012**: This agent uses a machine learning approach to select the negotiation strategy that is predicted to be most successful based on the structural features of the domain. [59].

For our second test setting, our agent *SAC Agent*, *Random Party* and *RLBOA Agent* negotiates with the ANAC 2011[49] finalist agents whose brief descriptions are given below:

- **HardHeaded**: The agent waits almost to the end of the negotiation deadline without conceding. Meanwhile, it tries to learn the opponent's preferences by analyzing the opponent's offers during the negotiation. By using

this information, the agent can find a suitable offer towards the end of the negotiation. The agent is the winner of ANAC 2011[14].

- **Gahboninho**: This agent aims to get a high score by putting deadline pressure on its opponent. In addition, The stubbornness of the agent is balanced by a behavior-based bidding model. This agent is ranked second in ANAC 2011[60].

- **IAMhaggler2011**: This agent uses a Gaussian process regression approach to estimate the concession moves of its opponents. Accordingly, it generates its offers. This agent is ranked third in ANAC 2011[61].

- **BRAM Agent**: The agent aims to reach an agreement by collaborating with the opponent. As time goes on, it searches for a win-win solution by conceding [62].

- **TheNegotiator**: This agent divides the negotiation into three phases and accordingly adjusts its concession speed. In general, it concedes slowly [63].

- **Nice Tic for Tat**: This agent mimics its opponent behavior to some extent by using Bayesian learning for opponent modeling. The agent follows a concession mechanism such that it reacts in opponent moves in a reciprocal direction and a proportionate amount [12].

- **AgentK2**: The agent tries to predict the opponent's potential offers and accordingly compromises towards the estimated maximum benefit at the end of the negotiation [64].

- **Value Model Agent**: This agent effectively models the opponent's preferences and accordingly explores the offered space to approximate the opponent's concession [65].

5.2 Sensitivity Analysis for SAC Agent

In order to see the effect of reward functions and behavior cloning, we trained three different version of the *SAC Agent* as follows:

- *SAC without behavior cloning using the proposed reward function (SAC with R+)*: We trained our SAC agent to include only the new reward approach but not behavior cloning.

- *SAC with behavior cloning using the standard reward function (SAC with BC-R)*: The standard reward approach typically used by RL agents in the negotiation domain is the agent's agreement utility. We trained our SAC agent with behavior cloning using the standard reward function shown in Equation 5.2.

$$R_{standard} = \begin{cases} U^{a}_{agreement} * 100 & \text{if agreement} \\ -1 & \text{no agreement} \end{cases}$$

- *SAC with behavior cloning using the proposed reward function (SAC with BC-R+)*: We trained our SAC agent by using our new reward approach, which includes not only our agreement utility but also the opponent agent's agreement utility value to the reward proportionally, together with the behavior cloning approach.

$$R+ = \begin{cases} \frac{(U^{a}_{agreement})^2 * 100}{U^{o}_{agreement}} & \text{if agreement} \\ -1 & \text{no agreement} \end{cases}$$

We adopt self-training approach to train aforementioned *SAC Agents* on *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

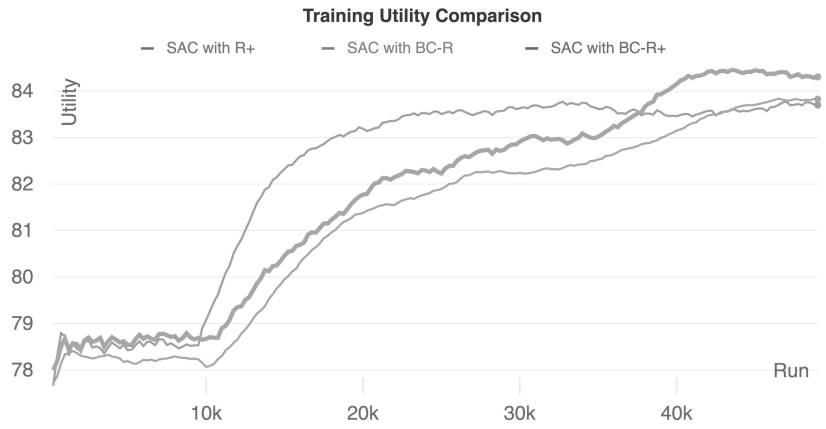


Figure 2: The moving average utility results of training session taken from *SAC* with BC, *SAC* without BC and SAC without new utility function

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*. Besides, it is observed that our proposed reward approach, which includes the utility of the opponent proportionally in reward, achieves higher results than the reward approach that only uses the agreement's utility. The agent converges later with our new reward approach but converges at higher scores compared to old reward approach.

5.3 Performance Evaluation

The performance of the *SAC Agent* is evaluated elaborately by comparing its performance with other agents considering to the agreement rate, received utility, winning rate and number of rounds to reach an agreement as explained below:

- ***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 receives a utility of zero. Note that the utilities ranges between zero and one in GENIUS.

- ***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 higher utility than or the same utility with the utility obtained by its opponent.

- ***Number of rounds to reach an agreement***: It corresponds to in which round that the agents reach an agreement. Agents may end up with an agreement early or late.

We analyzed the performance of the agent from three different perspectives for both settings. First of all, we observe the performance of the agent in the *party domain*, which was used for training in order to detect whether our agent is capable of negotiating with a variety of unknown opponent agents over a known domain (Section 5.3.1). Next, in order to see whether our agent is capable of negotiating irrespective of negotiation scenarios, we evaluate the performance of our agent negotiating with unknown opponent agents over unknown domains listed in Table 3. Here, we also investigate the effect of the domain size and that of the complexity of the negotiation scenarios in terms of opposition (Section 5.3.2). We compare the performance of *SAC Agent* with *RLBOA* and *Random Party* (Section 5.3.3). Lastly, we evaluate the performance of our agent with varying deadlines to see the effect of the deadline on its performance (Section 5.3.4).

5.3.1 Against Unknown Opponents on the *Party Domain*

In this section, we evaluate the performance of our agent against ANAC 2011 and 2012 finalist agents. For this comparison, The *SAC Agent* negotiates with each opponent 10 times on the *Party Domain* separately. We use two preference profiles per each domain that results in 20 negotiations with each opponent. In total, 160 negotiations take place between our agent and its opponent agents for each training setting.

First, we evaluate *SAC Agent*, which was trained with the ANAC 2011 finalists in a negotiation tournament including only the ANAC 2012 finalists. The average

agreement utility of our agent against those opponents is depicted in Figure 3 along with *RLBOA* and Random Party results. Note that we only consider the negotiation results which end with an agreement. It is obviously seen that *SAC Agent* receives higher agreement utility compared to RLBOA and Random Party except the case when it negotiated with *BramAgent 2*. Note that our agent could not reach an agreement with *BramAgent 2* at all on this domain. While *Sac Agent* has an average of 80 or more agreement utility, this figure is around 60-70 for *Random Party* and *RLBOA*.

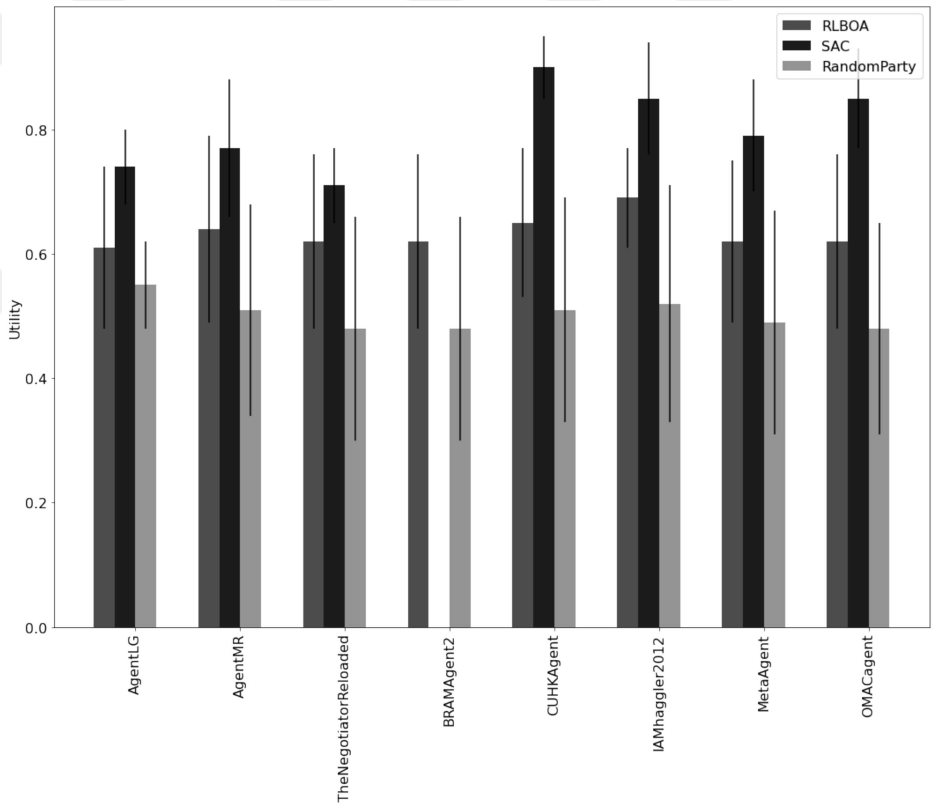


Figure 3: The performance comparison of *SAC Agent*, *RLBOA* and *Random Party* against the ANAC 2012 agents on the party domain

In addition, we examine the agreement rates, winning rates, and average agreement rounds of our agent listed in Table 4. It is seen that *SAC Agent* had difficulty in finding an agreement against tough agents such as AgentLG, BramAgent and CHUK Agent. When we found an agreement against these agents, our opponents

achieved higher utility than *SAC Agent* (i.e., low winning rate received in those negotiations). On the other hand, our agent has higher agreement rate when negotiation with AgentMR, IAMhaggler2012 and MetaAgent; however, AgentMR and MetaAgent beat *SAC Agent* most of the successful negotiations. We found out that agreement was reached at the last minutes while negotiating with competitive agents such as *AgentMR*, *MetaAgent* and *TheNegotiatorReloaded*.

Table 4: Comparison Against ANAC 2012 Successful Agents in Party Domain

	AgentLG	AgentMR	TheNegotiator Reloaded	BRAMAgent2	CUHKAgent	IAMhaggler2012	MetaAgent	OMACagent
Agreement Rate	25.0	91.67	50.0	0.0	25.83	100.0	70.0	50.0
Winning Rate	0.0	18.33	0.0	0.0	0.0	58.33	17.64	62.5
Avg Agreement Round	64.75	93.35	98.5	0.0	17.47	37.25	81.11	98.88

Furthermore, we evaluate the success of *SAC Agent*, which was trained with ANAC 2012 finalists on the party domain, in a negotiation tournament including only the ANAC 2011 finalists on party domain. Figure 4 shows the average agreement utility of our agent against those opponents along with RLBOA and Random Party. The *SAC Agent* performed substantially better than the Random Party and RLBOA (on average 0.8 versus 0.6–0.7).

We also review the average agreement rate, win rate, and average agreement rounds for the *SAC Agent* per each opponent in Table 5. In this setting, our agent seems to have high acceptance rate on average except the case while negotiating with the *KLH Agent*. When we examine the winning ratio, our agent beats most of its opponents at least half of the negotiations except *TheNegotiator*, *AgentK2* and *HardHeaded*. *SAC Agent* found the agreement in the last round when it negotiates with *ValueModelAgent* and *Gahboninho*.

Table 5: Comparison Against ANAC 2011 Successful Agents in Party Domain

	TheNegotiator	ValueModelAgent	Gahboninho	BRAMAgent	HardHeaded	NiceTitForTat	IAMhaggler2011	Agent_K2
Agreement Rate	100.0	100.0	50.0	90.0	0.0	100.0	90.0	80.0
Winning Rate	35.0	50.0	100.0	55.56	0	50.0	72.22	37.5
Avg Agreement Round	83.5	100.0	100.0	86.17	0	65.55	67.11	73.69

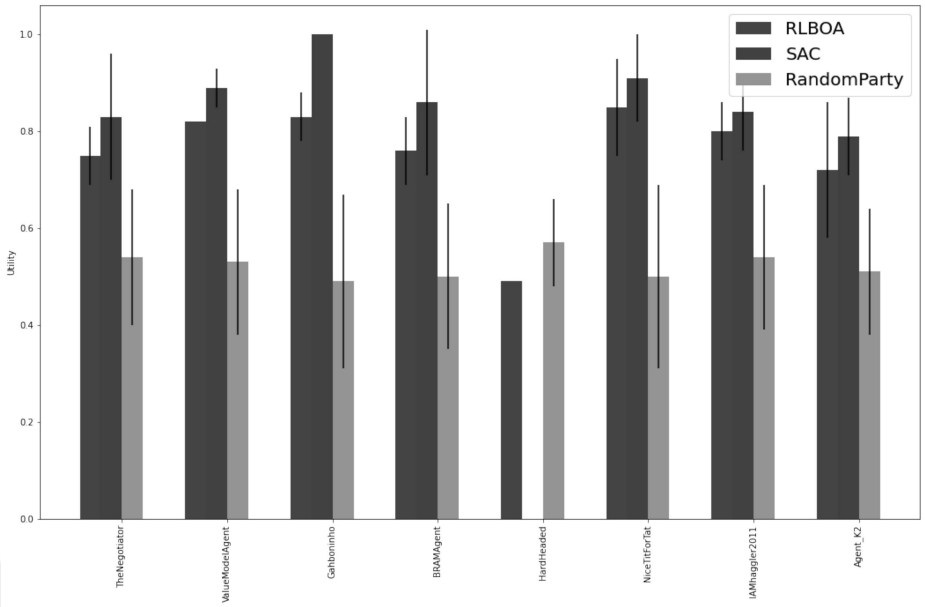


Figure 4: The performance comparison of *SAC Agent*, *RLBOA* and *Random Party* against the ANAC 2011 agents on party domain

5.3.2 Against Unknown Opponents on Unknown Domains

In this section, we study how well our agent would perform on the negotiation domains that have not been used during the training phase of the *SAC Agent*. For this purpose, for both experiment settings, we ran negotiation tournaments against ANAC 2011-2012 finalists involving a variety of negotiation domains selected from the ANAC competition described in Table 3. Each agent negotiates with their opponent 10 times for each negotiation scenario. That is, our agent performed 160 negotiations with each opponent (=2 profiles * 8 domains * 10 repetition). In total, 1280 negotiations took place between our agent and its opponents.

As a result of 1280 negotiations between the *SAC Agent* and the 2012 ANAC finalist agents, the *SAC Agent* is able to reach an agreement with its opponent in 529 of negotiations. Figure 5 shows the average utility of those agreements per each opponent. It is seen that benchmark agents generally have higher agreement utility than *SAC Agent*. On the other hand, *SAC Agent* obtained almost same or higher average utility when it negotiates with behavior-based negotiating agents such as *AgentMR* and *IAMhaggler2012*.

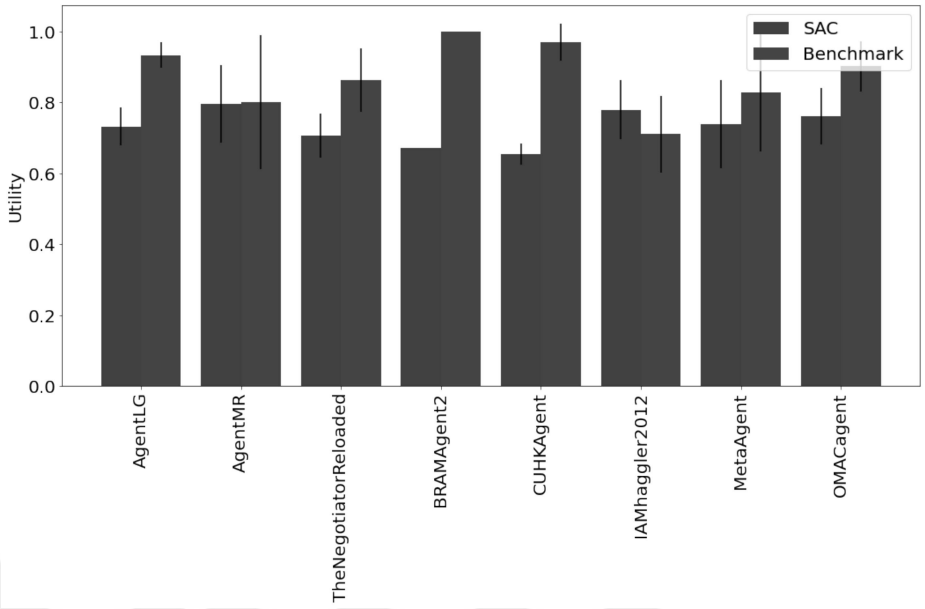


Figure 5: The average agreement results of our agent against ANAC 2012 agents over all test scenarios.

In order to analyze the performance of our agent in the second experiment setting, the *SAC Agent* negotiates with each ANAC 2011 finalist agent 10 times on the scenarios listed in Table 3 separately. In total, 1280 negotiations take place between our agent and the ANAC 2011 agents, in which 757 negotiation ended with an agreement. Among the negotiations, we consider the successful negotiation sessions ending with an agreement. For those successful negotiations, we separately analyze the negotiations per each opponent agent and demonstrate the average agent utility and opponent utility in Figure 6. 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 against the agents employing a more behavior-based

approach. Especially against *the Negotiator*, *Gahboninho*, *BRAM* and *NiceTitfotTat*, our agent shows an outstanding performance.

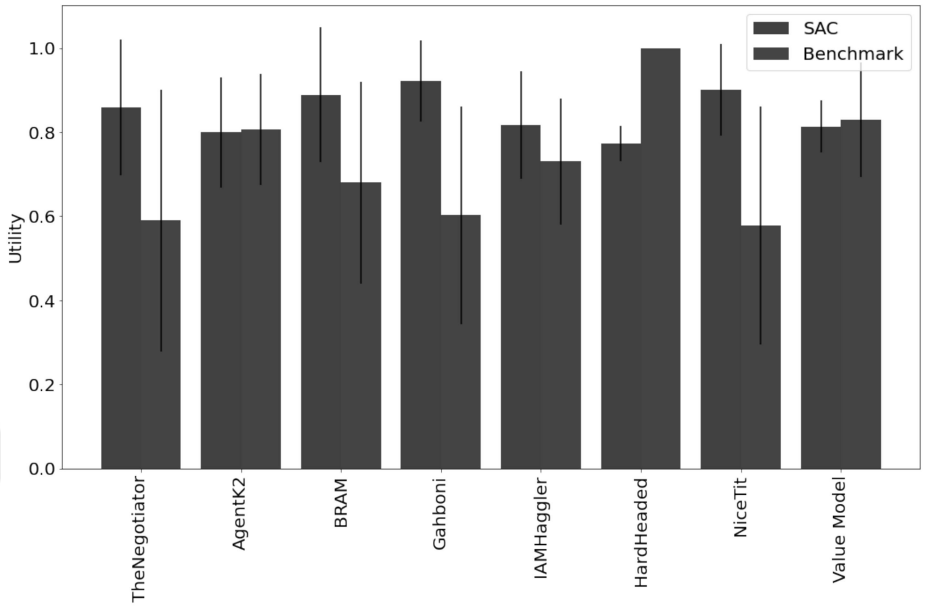


Figure 6: The average agreement results of our agent against ANAC 2011 agents over all test scenarios.

When we compare the results of these two test environments, it can be observed that our agent trained with the imitation data from the ANAC 2012 agents have more successful results. This may stem from the fact that the ANAC 2012 finalist agents are more competitive than the ANAC 2011 agents. Therefore, when we trained our agent with the data obtained from the negotiations against the ANAC 2012 agents and tested with the ANAC 2011 agents, our agent can beat some of those agents. Our agent might learn to negotiate competitively; thus, it may end up with a low agreement rate against competitive agents. However, when it negotiates with collaborative agents, it gains more utility.

5.3.3 Comparison with Random and RLBOA Agents

This section investigates 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 Party* and *RLBOA* as our baseline agents and

compare their performance with ours. We adopt the same negotiation setting in Section 5.3.2 for these baseline agents. They negotiate separately with the ANAC 2011 and ANAC 2012 finalists agents 10 times under the deadline of 100 rounds in the scenarios shown in Table 3.

Table 6 shows the average agreement rate and the average winning rate of the *SAC Agent* in the first test setting. Recall that in first setting we train our agent with self-learning by using a data taken from ANAC 2011 negotiation transactions. The most obvious observation we can make is that SAC agent performed worse than the Random Party and RLBOA agents in terms of agreement rate for almost all domains. Moreover, we observed that the average agreement rate of the *SAC Agent* drops when the size of the negotiation outcome space increases. In the Supermarket domain, which has a large domain and high opposition, SAC Agent had difficulty in finding agreement with its opponents.

Regarding the winning rate of *SAC Agent*, *RLBOA* and *Random Party*, it is seen that winning rate of our agent is higher than others while negotiating against all agents except *AgentLG*. It means that our agent acts more competitive than *RLBOA* and *Random Party* so that it can beat its opponent more often than they can. It is worth noting that the opponents – winners of the ANAC competition in our settings are also very competitive, which makes it difficult to beat them. For instance, none of these agents could beat *BramAgent2* and *CUHKAgent*.

Figure 7 shows the average utility of the agreements for *SAC Agent*, *RLBOA*, and *Random Party* when they reach an agreement with the ANAC 2012 agents on negotiation scenarios categorized with respect to size of the outcome space (small, medium, large and all). The *SAC Agent* achieved higher utility than both *RLBOA* and *Random Party* in small domains. On the other hand, as the domain size grew, the utility values of all compared agents began to decrease. Due to the *SAC agent* cannot manage to find an agreement against some opponents in medium and large outcome spaces, no results is visible for those opponents. Nevertheless, when our agent found agreement, it achieved much higher scores than *RLBOA* and *Random*

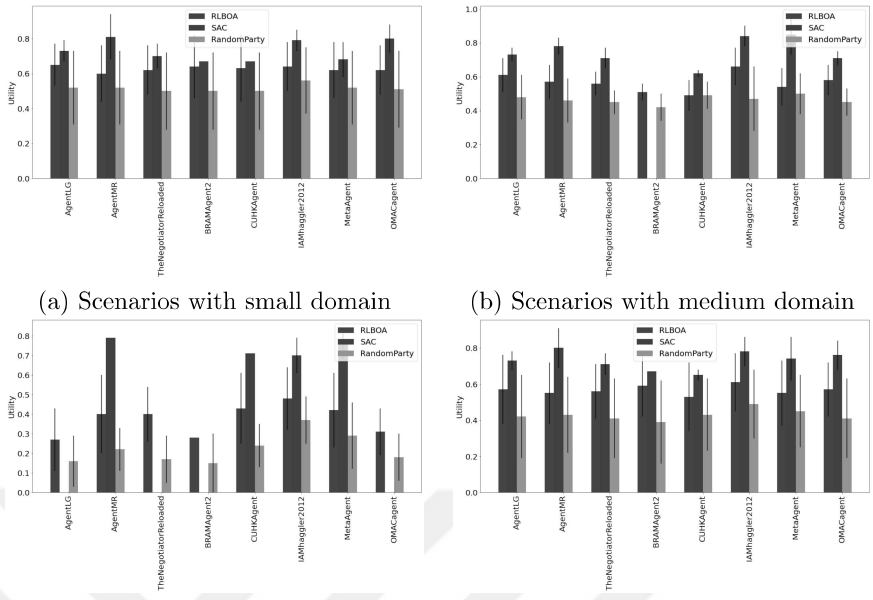
Table 6: Agreement Results of SAC, RLBOA, and Random Party Against ANAC 2012 Agents

		AgentLG		AgentMR		TheNegotiatorReloaded		BRAMAgent2		CUHKAgent		IAMhaggler2012		MetaAgent		OMACagent	
		AR	WR	AR	WR	AR	WR	AR	WR	AR	WR	AR	WR	AR	WR	AR	WR
Planes	SAC	50.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	50.0	50.0	30.0	25.0	100.0	50.0
	RLBOA	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	15.0	100.0	0.0	100.0	0.0
	Random	100.0	10.0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	20.0	100.0	0	100.0	0
Grocery	SAC	100.0	0	95.0	31.58	50.0	0	0.0	0	0.0	0	100.0	50.0	50.0	100.0	100.0	0
	RLBOA	100.0	0	100.0	0	100.0	0	50.0	0	50.0	0	100.0	0	100.0	0	100.0	0
	Random	100.0	0	100.0	5.0	100.0	0	100.0	0	100.0	0	100.0	5.0	100.0	0	100.0	0
Amsterdam	SAC	0.0	0	50.0	0	100.0	0	0.0	0	40.0	0	50.0	100.0	60.0	8.33	50.0	0
	RLBOA	100.0	0	100.0	0	100.0	0	50.0	0	50.0	0	100.0	0	65.0	0	100.0	0
	Random	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	5.0	100.0	0
Englandvs Zimbabwe	SAC	100.0	0	100.0	35.0	100.0	50.0	0.0	0	0.0	0	100.0	50.0	100.0	10.0	50.0	0
	RLBOA	100.0	0	100.0	0	100.0	0	15.0	0	15.0	0	100.0	0	80.0		100.0	0
	Random	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	5.0	100.0	0	100.0	0
AirportSite Selection	SAC	50.0	0	50.0	0	100.0	0	50.0	0	50.0	0	100.0	50.0	100.0	50.0	50.0	0
	RLBOA	100.0	0	100.0	0	100.0	0	70.0	0	70.0	0	100.0	0	100.0	0	100.0	0
	Random	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0
Supermarket	SAC	0.0	0	0.0	0	0.0	0	0.0	0	5.0	0	50.0	100.0	35.0	71.43	0.0	0
	RLBOA	60.0	0	95.0	0	85.0	0	5.0	0	100.0	0	100.0	0	85.0	0	55.0	0
	Random	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0
Itexvs Cypress	SAC	0.0	0	40.0	100.0	0.0	0	0.0	0	0.0	0	100.0	100.0	0.0	0	0.0	0
	RLBOA	0.0	0	25.0	0	10.0	0	20.0	0	25.0	0	95.0	5.26	30.0	0	10.0	0
	Random	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0
Energy Small	SAC	0.0	0	40.0	100.0	0.0	0	0.0	0	0.0	0	100.0	100.0	0.0	0	0.0	0
	RLBOA	45.0	0	55.0	0	25.0	0	0.0	0	75.0	0	100.0	5.0	80.0	18.75	10.0	0
	Random	100.0	0	95.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0	0
Small Domain	SAC	50.0	0.0	72.5	46.25	50.0	12.5	12.5	0.0	12.5	0.0	87.5	62.5	57.5	10.0	50.0	12.5
	RLBOA	75.0	0.0	81.25	0.0	77.5	0.0	51.25	0.0	52.5	0.0	98.75	5.14	77.5	0.0	77.5	0.0
	Random	100.0	2.5	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	6.25	100.0	0.0	100.0	0.0
Medium Domain	SAC	50.0	0.0	72.5	16.67	50.0	75.0	0.0	0.0	20.0	0.0	75.0	50.0	55.0	28.57	75.0	0.0
	RLBOA	100.0	0.0	100.0	0.0	100.0	0.0	50.0	0.0	50.0	0.0	100.0	0.0	82.5	0.0	100.0	0.0
	Random	100.0	0.0	100.0	2.5	100.0	0.0	100.0	0.0	100.0	0.0	100.0	2.5	100.0	2.5	100.0	0.0
Large Domain	SAC	0.0	0.0	20.0	25.0	0.0	0.0	0.0	0.0	2.5	0.0	75.0	75.0	17.5	41.67	0.0	0.0
	RLBOA	52.5	0.0	75.0	0.0	55.0	0.0	2.5	0.0	87.5	0.0	100.0	2.5	82.5	10.72	32.5	0.0
	Random	100.0	0.0	97.5	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0

Party in medium and large outcome spaces.

We also study how the performance of the *SAC Agent* can be influenced by the opposition of domain changes. Accordingly, as we described earlier, we grouped the domains with their opposition as low, medium and large and compared the negotiation results of the agents. According to this comparison, the results of our agent in the first setting are shown in Figure 8. It is seen that the success of the *SAC Agent* is the best for the domains with medium opposition compared to ones with low and high opposition. When there is a high level opposition in the given scenarios, our agent has difficulty in finding agreements as expected. In general, the varying opposition does not influence the performance of *SAC Agent* when it negotiates against *AgentMR* and *IAMhaggler2012* agents. Moreover, when it finds agreements, it achieve higher utility than *RLBOA* and *Random Party*. The average agreement utility drops significantly when they negotiate on domains with high opposition. The *SAC agent* is able to reach agreements with only *AgentMR*, *ChuckAgent*, *IAMhaggler2012* and *MetaAgent* agents; however, it receives higher scores than *RLBOA* and *Random Party* do by a large margin.

For the second test setting, Table 7 shows the average agreement rate and the



(a) Scenarios with small domain

(b) Scenarios with medium domain

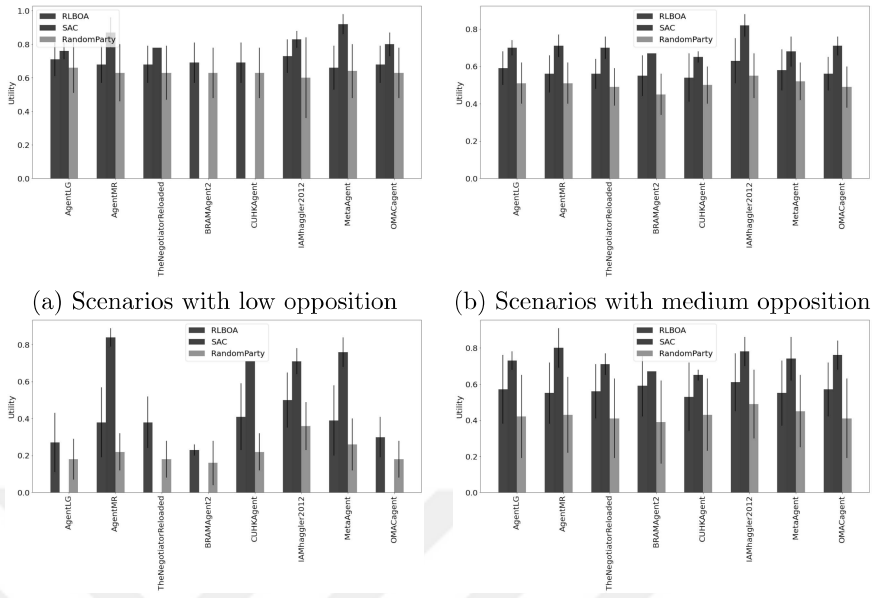
(c) Scenarios with large domains

(d) All scenarios

Figure 7: Average performance comparison of RL agent, RLBOA, and Random Party against ANAC 2012 agents in domains with small, medium, and large outcome spaces, respectively

average winning rate of the *SAC Agent*, *RLBOA* and *Random Party* against each opponent on the aforementioned negotiation scenarios. It is worth noting that the scenarios are sorted in increasing order with respect to their opposition level. It can be obviously observed that against *The Negotiator* and *Nice Tit-For-Tat* opponents, all agents have a high agreement rate. When the *SAC Agent* negotiates with *Value Model Agent* and *IAMhaggler2011*, the agreement rate has a tendency to decrease when the negotiation scenario becomes more competitive (i.e., higher opposition). The last part of the table shows the aggregated results of scenarios by grouping them with respect to the size of the outcome space. It can be seen that the average agreement rate of the *SAC Agent* drops in line with the size of the negotiation outcome space when it negotiates with the *BRAM Agent* and *Gahboninho*. As expected, the Random Party always ends up with an agreement (i.e., having a 100 per cent agreement rate). On average, *RLBOA* has a higher agreement rate than ours but its winning rate is lower than that of the *SAC Agent*.

While *RLBOA* and *Random Party* could not beat *Value Model Agent* at all,



(a) Scenarios with low opposition

(b) Scenarios with medium opposition

(c) Scenarios with high opposition

(d) All scenarios

Figure 8: Average performance comparison of RL Agent, RLBOA, and Random Party against ANAC 2012 agents in scenarios with low, medium, and high opposition, respectively

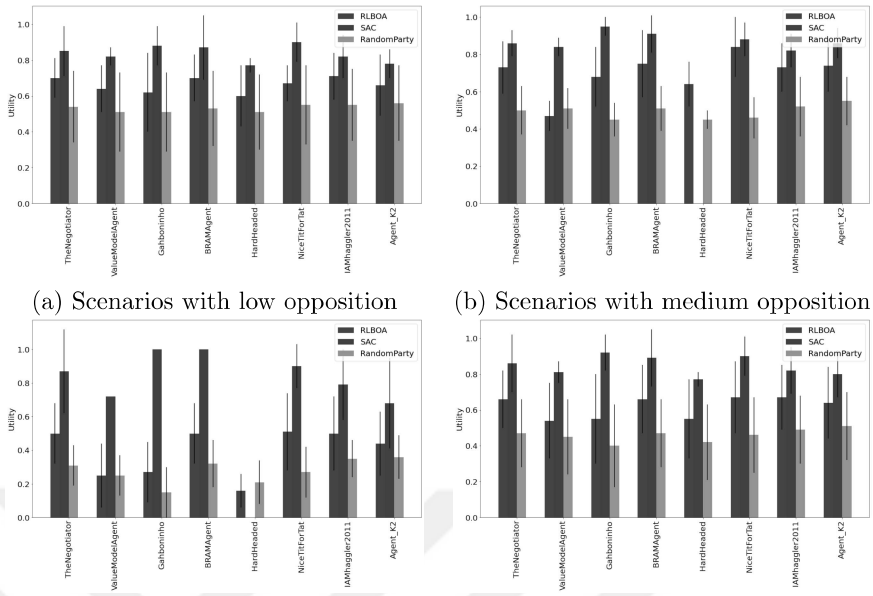
the *SAC Agent* manage to beat it in some scenarios. When we look at the average utilities, we could see the average utility difference is not high. That means, when the *Value Model Agent* beat us, it receives slightly higher utility than ours. Similarly, our winning rate is better than *Random Party* and *RLBOA* while negotiating with *Gahboninho*. In this case, our agent receives much higher utility when it beats the *Gahboninho* as seen in Figure 9. When we look at the log files elaborately, we noticed that they reached an agreement towards the end of the negotiation. Our winning rate is quite high against *The Negotiator*, *IAMhaggler2011* and *Nice Tic For Tat*. On the other hand, our winning rates are relatively low against the *HardHeaded* and *Agent K2*. Those agents mostly do not concede until the very last moment of the negotiation. As expected, the winning rate of the *Random Party* is extremely low while that of *RLBOA* is slightly better but still lower than ours.

Next, we categorize all negotiation scenarios in terms of the size of the domain and opposition of the scenario and report the results in Table 8 elaborately. Note that best-performing values are boldface in the table. First, it can be observed

Table 7: Average Agreement Results(AR) and Average Winning Results(WR) of SAC, RLBOA and Random Party Againts ANAC 2011 Agents

		TheNegotiator		ValueModelAgent		Gahboninho		BRAMAgent		HardHeaded		NiceTitForTat		IAMhaggler2011		Agent_K2	
		AR	**WR**	**AR**	**WR**	**AR**	**WR**	**AR**	**WR**	**AR**	**WR**	**AR**	**WR**	**AR**	**WR**	**AR**	**WR**
Planes	**SAC**	95.0	38.89	90.0	0.0	90.0	0.0	95.0	11.11	95.0	0.0	90.0	0.0	90.0	100.0	90.0	5.56
	RLBOA	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	35.0	100.0	25.0
	Random	100.0	5.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	15.0	100.0	20.0	100.0	15.0
Grocery	**SAC**	100.0	60.0	100.0	0.0	50.0	50.0	100.0	60.0	0.0	0.0	100.0	65.0	95.0	59.44	80.0	62.5
	RLBOA	100.0	0.0	20.0	0.0	50.0	15.0	100.0	45.0	50.0	0.0	100.0	75.0	100.0	0.0	100.0	5.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	10.0	100.0	5.0
Amsterdam	**SAC**	100.0	85.0	50.0	50.0	50.0	50.0	5.0	0.0	0.0	0.0	100.0	65.0	90.0	61.25	95.0	89.44
	RLBOA	100.0	15.0	25.0	0.0	40.0	0.0	100.0	20.0	30.0	0.0	100.0	55.0	100.0	5.0	100.0	15.0
	Random	100.0	5.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Englandvs Zimbabwe	**SAC**	100.0	85.0	50.0	50.0	50.0	50.0	100.0	100.0	0.0	0.0	100.0	95.0	100.0	15.0	35.0	25.0
	RLBOA	100.0	0.0	75.0	0.0	50.0	0.0	100.0	25.0	80.0	0.0	100.0	5.0	100.0	5.0	100.0	0.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	5.0	100.0	0.0
AirportSite Selection	**SAC**	100.0	50.0	50.0	50.0	50.0	50.0	100.0	45.0	0.0	0.0	100.0	95.0	95.0	57.22	75.0	60.0
	RLBOA	100.0	0.0	70.0	0.0	50.0	0.0	100.0	30.0	100.0	0.0	100.0	0.0	100.0	40.0	100.0	10.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Supermarket	**SAC**	100.0	100.0	50.0	50.0	50.0	50.0	40.0	50.0	0.0	0.0	100.0	100.0	55.0	100.0	30.0	50.0
	RLBOA	100.0	0.0	40.0	0.0	40.0	0.0	100.0	0.0	15.0	0.0	100.0	10.0	80.0	0.0	95.0	0.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Itexvs Cypress	**SAC**	100.0	100.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	100.0	95.0	50.0	100.0	5.0	0.0
	RLBOA	100.0	5.0	75.0	0.0	35.0	0.0	100.0	30.0	35.0	0.0	100.0	0.0	80.0	0.0	75.0	0.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Energy Small	**SAC**	100.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	10.0	50.0	20.0	25.0
	RLBOA	100.0	0.0	70.0	0.0	50.0	0.0	100.0	30.0	100.0	0.0	100.0	0.0	100.0	40.0	100.0	10.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Small Domain	**SAC**	98.75	68.47	47.5	25.0	47.5	25.0	76.25	39.03	23.75	0.0	97.5	71.25	83.75	68.06	51.25	22.64
	RLBOA	100.0	1.25	81.25	0.0	81.25	0.0	100.0	21.25	81.25	0.0	100.0	1.25	95.0	20.0	92.5	8.75
	Random	100.0	1.25	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	3.75	100.0	6.25	100.0	3.75
Medium Domain	**SAC**	100.0	72.5	75.0	25.0	50.0	50.0	52.5	30.0	0.0	0.0	100.0	65.0	92.5	60.35	87.5	75.97
	RLBOA	100.0	7.5	75.0	0.0	100.0	7.5	100.0	32.5	75.0	0.0	100.0	65.0	100.0	2.5	100.0	10.0
	Random	100.0	2.5	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	5.0	100.0	2.5
Large Domain	**SAC**	100.0	85.0	25.0	25.0	25.0	25.0	20.0	25.0	0.0	0.0	55.0	50.0	32.5	75.0	25.0	37.5
	RLBOA	100.0	0.0	55.0	0.0	45.0	0.0	100.0	15.0	57.5	0.0	100.0	5.0	90.0	20.0	97.5	5.0
	Random	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0

that the winning rate of the *SAC Agent* is much higher than *RLBOA* and *Random Party* 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 percent). 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 Party* (approximately 85 versus 60-73 and 27-53 respectively). To study the effect of the domains elaborately, the negotiation sessions are grouped with respect to the domain size (small, medium, large) and the average utility received by the baseline agents and *SAC Agent* while negotiating against each opponent strategy is shown in Figure 9. Except against the *HardHeaded*, our agent outperforms others with respect to the received utility. Interestingly, the utility gained by the *Random Party* and *RLBOA* falls down significantly when the size of the domain becomes larger. However, we did not observe such a decrease for the *SAC Agent*. Note that the fourth graph shows the average utilities when we consider all negotiation domains.



(a) Scenarios with low opposition

(b) Scenarios with medium opposition

(c) Scenarios with high opposition

(d) All scenarios

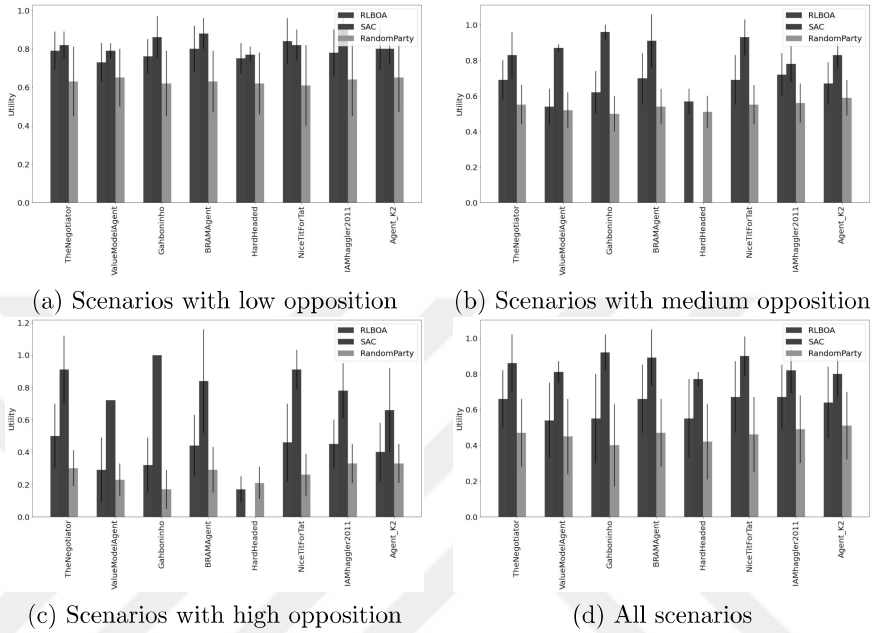
Figure 9: Average performance comparison of SAC Agent, RLBOA, and Random Party against ANAC 2011 agents in domains with small, medium, and large outcome spaces, respectively

Table 8: Comparison of the *SAC Agent*, *RLBOA* and *Random Party* in negotiations grouped by domain size.

	Small Domains			Medium Domains			Large Domains			Low Opposition			Medium Opposition			High Opposition		
	SAC	RLBOA	Random	SAC	RLBOA	Random	SAC	RLBOA	Random	SAC	RLBOA	Random	SAC	RLBOA	Random	SAC	RLBOA	Random
Winning Rate	**56.77**	8.89	1.88	**65.47**	20.58	1.25	**89.38**	6.98	0.0	**35.29**	14.08	4.38	**75.24**	13.46	0.42	**90.36**	8.12	0.0
Agreement Rate	65.78	84.38	**100.0**	69.69	75.94	**100.0**	35.31	80.62	**100.0**	85.0	88.75	**100.0**	66.46	78.96	**100.0**	34.58	79.58	**100.0**
Acceptance Utility	**84.66**	67.21	53.18	**86.79**	73.37	49.26	**85.77**	62.42	27.75	**83.49**	78.57	63.15	**86.49**	66.65	53.79	**86.68**	40.7	26.36
Acceptance Count	421	540	**640**	223	243	**320**	113	258	**320**	272	284	**320**	319	379	**480**	166	382	**480**
Winning Count	**239**	48	12	**146**	50	4	**101**	18	0	**96**	40	14	**240**	51	2	**150**	31	0
Total Negotiation Count	640	640	640	320	320	320	320	320	320	320	320	320	480	480	480	480	480	480

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 which 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 are given against the ANAC 2011 agents are given in Figure 10. 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 for its even in competitive scenarios.



(a) Scenarios with low opposition

(b) Scenarios with medium opposition

(c) Scenarios with high opposition

(d) All scenarios

Figure 10: Average performance comparison of SAC agent, RLBOA, and Random Party 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 Random Party and RLBOA agents (approximately 64% versus 2% and 11%, respectively). That is, although *Random Party* 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). Moreover, on average the *RLBOA* reaches agreement sooner than *Random Party* and the *SAC Agent* (around 54 rounds versus 64 and 84 rounds, respectively). In this respect, we can conclude that our agent might be more patient than other agents to wait towards the end of the negotiation to reach an agreement with a high utility. The average utility scores support this observation.

Table 9: Comparison against ANAC 2012 Agents in different deadlines on Party Domain

	Deadline 50 Round		Deadline 100 Round		Deadline 150 Round		Deadline 200 Round	
	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility
AgentLG	0.72 ± 0.0	0.98 ± 0.0	0.72 ± 0.0	0.98 ± 0.0	0.72 ± 0.0	0.98 ± 0.0	0.72 ± 0.0	0.98 ± 0.0
AgentMR	0.76 ± 0.08	0.92 ± 0.04	0.7 ± 0.08	0.88 ± 0.04	0.7 ± 0.05	0.94 ± 0.04	0.68 ± 0.02	0.93 ± 0.07
TheNegotiatorReloaded	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0
BRAMAgent2	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0
CUHKAgent	0.72 ± 0.0	0.96 ± 0.01	0.7 ± 0.03	0.97 ± 0.01	0.68 ± 0.04	0.98 ± 0.02	0.7 ± 0.03	0.96 ± 0.01
IAMhaggler2012	0.84 ± 0.03	0.74 ± 0.01	0.74 ± 0.03	0.84 ± 0.03	0.82 ± 0.11	0.76 ± 0.11	0.89 ± 0.05	0.75 ± 0.08
MetaAgent	0.7 ± 0.06	0.92 ± 0.09	0.71 ± 0.07	0.92 ± 0.09	0.72 ± 0.07	0.92 ± 0.09	0.7 ± 0.07	0.94 ± 0.08
OMACagent	0.76 ± 0.1	0.92 ± 0.08	0.79 ± 0.07	0.91 ± 0.06	0.76 ± 0.1	0.92 ± 0.08	0.69 ± 0.03	0.94 ± 0.06

5.3.4 Studying the Effect of Deadline

In order to study the effect of the deadline on the performance of the *SAC agents*, we ran the same negotiation setting with varying deadlines (50, 100, 150, and 200 rounds) for both test settings. Recall that the *SAC Agent* is trained under the deadline of 100 rounds.

Figure 11 shows the average agreement utility of the *SAC Agent* per each opponent for different deadlines in the first setting. It can be seen that the performance of the *SAC Agent* does not change with respect to the varying deadline when it negotiates with *AgentLG*, *TheNegotiationReloaded*, *BRAMAgent2* agents. Moreover, it received better agreements when it negotiates with *CUHCKAgent* and *AgentMR* in a deadline of 50 rounds compared to other deadlines. Interestingly, it ends up with a worse negotiation agreement against *AgentMR* when the deadline is longer. The average agreement utility varies on different deadlines when it negotiates with the *CUHCKAgent* and *IAMHaggler2012*. Furthermore, we can analyze the effect of the deadline in terms of the average agreement utility scores of *SAC agent* and the ANAC 2012 agents as shown in Table 9. It can clearly seen that our agent trained with the first setting obtain divergent results against the ANAC 2012 agents on varying deadlines.

Figure 12 shows the average agreement utility of our agent for varying deadlines for the second setting. It seems that the performance of our agent for most of the cases is not affected significantly by the deadline. On one hand, it achieves slightly better agreements against *TheNegotiatior* and *Agent_K2* when they have a short deadline such as 50 rounds. On the other hand, it fails to reach an agreement

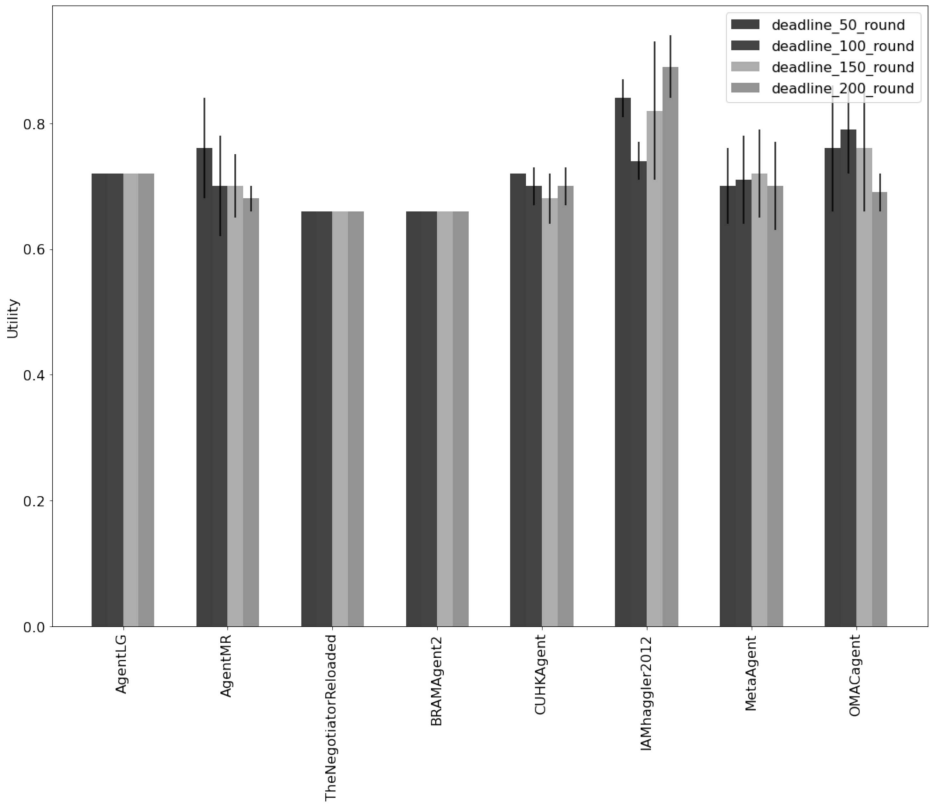


Figure 11: The performance graph of *SAC Agent* in *party domain* against ANAC 2012 agents with different deadlines

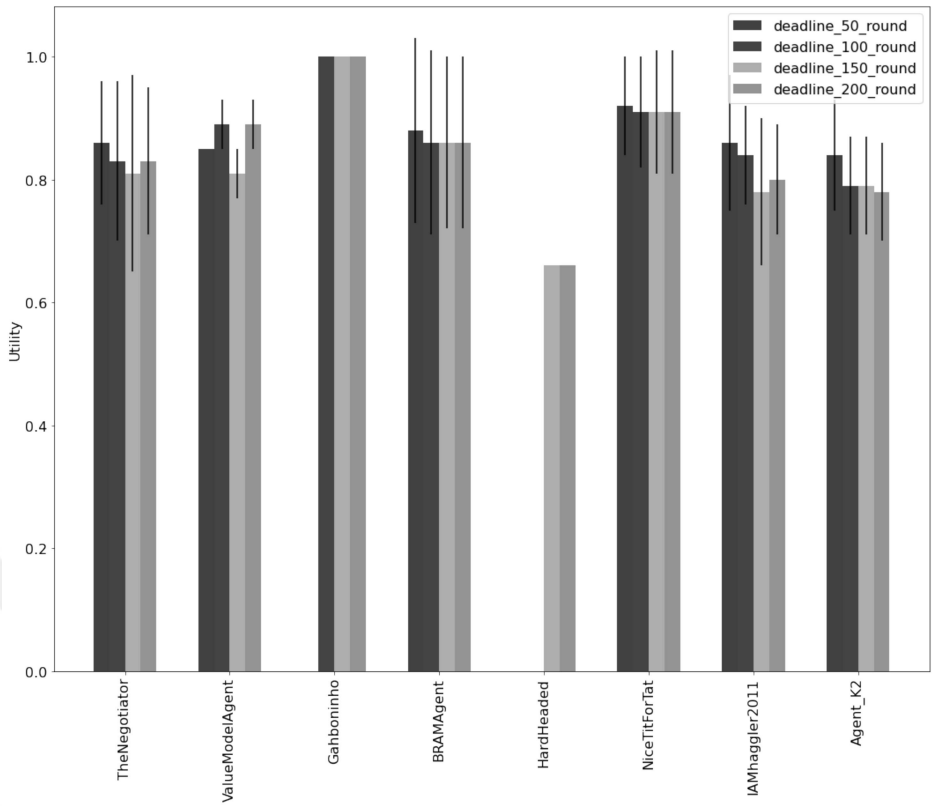


Figure 12: The performance graph of *SAC Agent* in *party domain* against ANAC 2011 agents with different deadlines

while negotiating against *Gahboninho* and *HardHeaded* on short deadlines such as 50 rounds. Recall that our agent is trained only with the deadline of 100 rounds but it can perform well for different deadlines except the aforementioned cases.

Furthermore, we can analyze the effect of the deadline on the agreement utility for the *SAC Agent* and opponent agents as shown in Table 10. On average, our agent has succeeded in competitive results against its competitors in all deadlines. As the deadline gets longer, some agents such as *Negotiator* and *Agent_K2*, obtained better agreements for themselves, resulting in a decrease on the utility of the *SAC Agent*.

Table 10: Comparison against ANAC 2011 Agents in different deadlines on Party Domain

	Deadline 50 Round		Deadline 100 Round		Deadline 150 Round		Deadline 200 Round	
	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility	SAC Agent Utility	Opponent Agent Utility
TheNegotiator	0.86 ± 0.1	0.8 ± 0.1	0.83 ± 0.13	0.81 ± 0.12	0.81 ± 0.16	0.81 ± 0.12	0.83 ± 0.12	0.83 ± 0.12
ValueModelAgent	0.85 ± 0.0	0.86 ± 0.0	0.89 ± 0.04	0.82 ± 0.04	0.81 ± 0.04	0.84 ± 0.02	0.89 ± 0.04	0.82 ± 0.04
Gahboninho	0	0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0	1.0 ± 0.0	0.66 ± 0.0
BRAMAgent	0.88 ± 0.15	0.78 ± 0.15	0.86 ± 0.15	0.8 ± 0.16	0.86 ± 0.14	0.81 ± 0.15	0.86 ± 0.14	0.8 ± 0.15
HardHeaded	0	0	0	0 ± 0	0.66 ± 0.0	0.92 ± 0.0	0.66 ± 0.0	0.92 ± 0.0
IAMhaggler2011	0.92 ± 0.08	0.76 ± 0.1	0.91 ± 0.09	0.76 ± 0.1	0.91 ± 0.1	0.76 ± 0.11	0.91 ± 0.1	0.77 ± 0.11
NiceTitForTat	0.86 ± 0.11	0.76 ± 0.07	0.84 ± 0.08	0.77 ± 0.09	0.78 ± 0.12	0.8 ± 0.05	0.8 ± 0.09	0.81 ± 0.06
Agent_K2	0.84 ± 0.09	0.78 ± 0.1	0.79 ± 0.08	0.86 ± 0.09	0.79 ± 0.08	0.88 ± 0.08	0.78 ± 0.08	0.89 ± 0.06

CHAPTER VI

CONCLUSION

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 2011-2012 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 2011-2012 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 when there is an agreement. However, there is a room for improvement in terms of agreement rate. Our agent mostly negotiate competitively; therefore, it has difficulty in reaching agreements with other competitive agents. In addition, we analyze our agents in a variety of scenarios with varying domain size 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).

As future work, the behavior cloning structure of our agent can be further revised to make it superior to the rest of the agents. In addition, it would be interesting to employ our agent in human-agent negotiations to see how well it can negotiate with human negotiators.

Bibliography

[1] B. An, V. Lesser, and K. M. Sim, "Strategic agents for multi-resource negotiation," *Autonomous Agents and Multi-Agent Systems*, vol. 23, pp. 114–153, Jul 2011.

[2] C. Eran, M. O. Keskin, F. Cantürk, and R. Aydoğan, "A decentralized token-based negotiation approach for multi-agent path finding," in *Multi-Agent Systems* (A. Rosenfeld and N. Talmon, eds.), (Cham), pp. 264–280, Springer International Publishing, 2021.

[3] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. J. Wooldridge, and C. Sierra, "Automated negotiation: prospects, methods and challenges," *Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, 2001.

[4] T. Sandholm, "Agents in electronic commerce: Component technologies for automated negotiation and coalition formation," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 1, pp. 73–96, 2000.

[5] S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2, pp. 1234–1239, IEEE, 1999.

[6] R. Aydoğan, P. Øzturk, and Y. Razeghi, "Negotiation for incentive driven privacy-preserving information sharing," in *PRIMA 2017: Principles and Practice of Multi-Agent Systems* (B. An, A. Bazzan, J. Leite, S. Villata, and L. van der Torre, eds.), (Cham), pp. 486–494, Springer International Publishing, 2017.

[7] T. Baarslag, E. H. Gerding, R. Aydogan, and M. C. Schraefel, "Optimal Negotiation Decision Functions in Time-Sensitive Domains," in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 2, pp. 190–197, 2015.

[8] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation decision functions for autonomous agents," *Robotics and Autonomous Systems*, vol. 24, no. 3-4, pp. 159–182, 1998.

[9] R. Aydoğan, O. Kafalı, F. Arslan, C. M. Jonker, and M. P. Singh, "Nova: Value-based Negotiation of Norms," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, pp. 1–29, Aug. 2021.

[10] I. Marsa-Maestre, M. Klein, C. M. Jonker, and R. Aydoğan, "From problems to protocols: Towards a negotiation handbook," *Decision Support Systems*, vol. 60, pp. 39–54, 2014.

[11] J.-G. Cao, "Research on electronic commerce automated negotiation in multi-agent system based on reinforcement learning," in *2009 International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1419–1423, 2009.

[12] T. Baarslag, K. Hindriks, and C. Jonker, *A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations*, pp. 229–233. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[13] S. Morii and T. Ito, *AgentMR: Concession Strategy Based on Heuristic for Automated Negotiating Agents*, pp. 181–185. Tokyo: Springer Japan, 2014.

[14] T. Krimpen, D. Looije, and S. Hajizadeh, *HardHeaded*, vol. 435, pp. 223–227. Springer, 01 2013.

[15] W. Han, "Sellers' pricing by bayesian reinforcement learning," in *2009 International Conference on E-Business and Information System Security*, pp. 1–5, 2009.

[16] V. Sunder, L. Vig, A. Chatterjee, and G. Shroff, "Prosocial or selfish? agents with different behaviors for contract negotiation using reinforcement learning," *In Advances in Automated Negotiations*, pp. 63–81, 01 2018.

[17] P. Bagga, N. Paoletti, B. Alrayes, and K. Stathis, "A deep reinforcement learning approach to concurrent bilateral negotiation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pp. 297–303, 2020.

[18] A. Sengupta, Y. Mohammad, and S. Nakadai, "An autonomous negotiating agent framework with reinforcement learning based strategies and adaptive strategy switching mechanism," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1163–1172, 2021.

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

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, 02 2015.

[21] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[22] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*, (Berlin, Heidelberg), pp. 363–372, Springer Berlin Heidelberg, 2006.

[23] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, p. 1334–1373, Jan. 2016.

[24] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search.," in *ICRA*, pp. 3381–3388, IEEE, 2017.

[25] G. Tesauro and J. Kephart, "Pricing in agent economies using multi-agent q-learning," *Autonomous Agent and Multi-Agent Systems*, vol. 5, 10 1999.

[26] M. Sridharan and G. Tesauro, *Multi-agent Q-learning and Regression Trees for Automated Pricing Decisions*, pp. 217–234. Boston, MA: Springer US, 2002.

[27] A. Papangelis and K. Georgila, "Reinforcement learning of multi-issue negotiation dialogue policies," in *SIGDIAL Conference*, pp. 154–158, 2015.

[28] Y. Zou, W. Zhan, and Y. Shao, "Evolution with reinforcement learning in negotiation," *PLOS ONE*, vol. 9, pp. 1–7, 07 2014.

[29] M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra, "Deal or no deal? end-to-end learning of negotiation dialogues," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 2443–2453, Association for Computational Linguistics, Sept. 2017.

[30] J. Jin, C. Song, H. Li, K. Gai, J. Wang, and W. Zhang, "Real-time bidding with multi-agent reinforcement learning in display advertising," *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[31] D. Kröhling, O. Chiotti, and E. Martínez, "The importance of context-dependent learning in negotiation agents," *Inteligencia Artif.*, vol. 22, pp. 135–149, 2019.

[32] J. Bakker, A. Hammond, D. Bloembergen, and T. Baarslag, "Rlboa: A modular reinforcement learning framework for autonomous negotiating agents," in *AAMAS*, 2019.

[33] Y. Razeghi, O. Yavuz, and R. Aydogan, "Deep reinforcement learning for acceptance strategy in bilateral negotiations," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, pp. 1824–1840, 2020.

[34] P. Bagga, N. Paoletti, B. Alrayes, and K. Stathis, "Anegma: an automated negotiation model for e-markets," *Autonomous Agents and Multi-Agent Systems*, vol. 35, p. 27, Jun 2021.

[35] T. D. Güneş, E. Arditi, and R. Aydoğan, "Collective Voice of Experts in Multilateral Negotiation," in *PRIMA 2017: Principles and Practice of Multi-Agent Systems* (B. An, A. Bazzan, J. Leite, S. Villata, and L. van der Torre, eds.), (Cham), pp. 450–458, Springer International Publishing, 2017.

[36] R. Aydoğan, D. Festen, K. V. Hindriks, and C. M. Jonker, "Alternating offers protocols for multilateral negotiation," in *Modern Approaches to Agent-based Complex Automated Negotiation*, pp. 153–167, Springer, 2017.

[37] A. Rubinstein, "The electronic mail game: Strategic behavior under "almost common knowledge"," in *Knowledge, Belief, and Strategic Interaction* (C. Bicchieri and M. L. Dalla Chiara, eds.), pp. 317–326, Cambridge, United Kingdom: Cambridge University Press, 1992.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[40] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, pp. 293–321, May 1992.

[41] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *ArXiv*, vol. abs/1812.05905, 2018.

[42] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, p. 2094–2100, AAAI Press, 2016.

[43] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[44] C. M. Jonker, R. Aydoğan, T. Baarslag, K. Fujita, T. Ito, and K. V. Hindriks, "Automated negotiating agents competition (ANAC)," in *Proceedings of the 31st Conference on Artificial Intelligence (AAAI)*, (California, USA), pp. 5070–5072, AAAI Press, 2017.

[45] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, vol. abs/1312.5602, 2013.

[46] C. M. Jonker, R. Aydoğan, T. Baarslag, K. Fujita, T. Ito, and K. V. Hindriks, "Automated negotiating agents competition (ANAC)," in *Proceedings of the 31st Conference on Artificial Intelligence (AAAI)*, (California, USA), pp. 5070–5072, AAAI Press, 2017.

[47] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2018.

[48] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. Hindriks, and C. M. Jonker, "Genius: An integrated environment for supporting the design of generic automated negotiators," *Computational Intelligence*, vol. 30, no. 1, pp. 48–70, 2014.

[49] K. Fujita, T. Ito, T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin, "The second automated negotiating agents competition (anac2011)," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*, pp. 183–197, Springer, 2013.

[50] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings, "An overview of the results and insights from the third automated negotiating agents competition (anac2012)," in *Novel Insights in Agent-based Complex Automated Negotiation*, pp. 151–162, Springer, 2014.

[51] J. Hao and H.-f. Leung, *CUHKAgent: An Adaptive Negotiation Strategy for Bilateral Negotiations over Multiple Items*, vol. 535, pp. 171–179. Springer, 01 2014.

[52] S. Chen and G. Weiss, "Omac: A discrete wavelet transformation based negotiation agent," *Studies in Computational Intelligence*, vol. 535, pp. 187–196, 01 2014.

[53] A. Dirkzwager and M. Hendrikx, "An adaptive negotiation strategy for real-time bilateral negotiations," *Studies in Computational Intelligence*, vol. 535, pp. 163–170, 01 2014.

[54] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 807–814, Omnipress, 2010.

[55] S. M. LaValle and M. S. Branicky, *On the Relationship between Classical Grid Search and Probabilistic Roadmaps*, pp. 59–75. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[56] K. Kim, C. Lee, E. Jeong, J. Han, M. Kim, C. Yoon, M. Kim, and J. Park, "Medipixel rl algorithms," 2020.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[58] T. Baarslag, M. Hendrikx, K. Hindriks, and C. Jonker, "Predicting the performance of opponent models in automated negotiation," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 2, pp. 59–66, IEEE, 2013.

[59] L. Ilany and Y. K. Gal, *The Simple-Meta Agent*, pp. 197–200. Tokyo: Springer Japan, 2014.

[60] M. Ben Adar Bessos, N. Sofy, and A. Elimelech, *Gahboninho: Strategy for Balancing Pressure and Compromise in Automated Negotiation*, vol. 435, pp. 205–208. Springer, 01 2013.

[61] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings, *IAMhaggler2011: A Gaussian Process Regression Based Negotiation Agent*, pp. 209–212. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[62] R. Fishel, M. Bercovitch, and Y. K. Gal, *BRAM Agent*, pp. 213–216. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[63] A. Dirkzwager, M. Hendrikx, and J. Ruiter, *TheNegotiator: A Dynamic Strategy for Bilateral Negotiations with Time-Based Discounts*, vol. 435, pp. 217–221. Springer, 01 2013.

[64] S. Kawaguchi, K. Fujita, and T. Ito, *AgentK2: Compromising Strategy Based on Estimated Maximum Utility for Automated Negotiating Agents*, vol. 383, pp. 235–241. Springer, 01 2013.

[65] A. Frieder and G. Miller, "Value model agent: A novel preference profiler for negotiation with agents.," in *Complex Automated Negotiations* (T. Ito, M. Zhang, V. Robu, and T. Matsuo, eds.), vol. 435 of *Studies in Computational Intelligence*, pp. 199–203, Springer, 2013.

[66] Y. Xu, J. Yao, H.-A. Jacobsen, and H. Guan, "Cost-efficient negotiation over multiple resources with reinforcement learning," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–6, IEEE, 2017.

[67] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 05 1992.

[68] C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford, 1989.

[69] K. Schmid., L. Belzner., T. Phan., T. Gabor., and C. Linnhoff-Popien., "Multi-agent reinforcement learning for bargaining under risk and asymmetric information," in *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,*, pp. 144–151, INSTICC, SciTePress, 2020.

[70] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[71] F. Oliveira, "Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning," *Omega*, vol. 47, 10 2013.

[72] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, p. 2094–2100, AAAI Press, 2016.