RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents

Jasper Bakker*
University of Amsterdam
Amsterdam, The Netherlands
jcsbakker@gmail.com

Aron Hammond*
University of Amsterdam
Amsterdam, The Netherlands
a.o.m.hammond@gmail.com

Daan Bloembergen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
d.bloembergen@cwi.nl

Tim Baarslag
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
t.baarslag@cwi.nl

ABSTRACT

Negotiation is a complex problem, in which the variety of settings and opponents that may be encountered prohibits the use of a single predefined negotiation strategy. Hence the agent should be able to learn such a strategy autonomously. To this end we propose RLBOA, a modular framework that facilitates the creation of autonomous negotiation agents using reinforcement learning. The framework allows for the creation of agents that are capable of negotiating effectively in many different scenarios. To be able to cope with the large size of the state and action spaces and diversity of settings, we leverage the modular BOA-framework. This decouples the negotiation strategy into a Bidding strategy, an Opponent model and an Acceptance condition. Furthermore, we map the multidimensional contract space onto the utility axis which enables a compact and generic state and action description. We demonstrate the value of the RLBOA framework by implementing an agent that uses tabular Q-learning on the compressed state and action space to learn a bidding strategy. We show that the resulting agent is able to learn well-performing bidding strategies in a range of negotiation settings and is able to generalize across opponents and domains.

KEYWORDS

Bargaining and negotiation; Learning agent-to-agent interactions (negotiation, trust, coordination); Reinforcement Learning

ACM Reference Format:
Jasper Bakker, Aron Hammond, Daan Bloembergen, and Tim Baarslag. 2019. RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 9 pages.

1 INTRODUCTION

Negotiation is omnipresent in today's society. From buying a car or produce on the market, to negotiations that impact people on a world scale, such as global carbon emission agreements. Automation of negotiation can offer a number of benefits compared to non-computerized negotiation. Autonomous negotiators can systematically consider all possible outcomes in a fraction of the time and for a fraction of the cost [10]. Negotiating agents can thus reduce the time and effort needed to reach agreements and simultaneously increase the chance of better, win-win, deals [5, 16].

Automating negotiation is a complex problem, with many open challenges left to be tackled that form important obstacles for a broader adoption of negotiating agents in real world settings [5]. Firstly, there are many different negotiation settings, and an agent that performs well in one domain or against one opponent may be useless in another setting. Secondly, when considering multi-issue negotiations, the state space increases exponentially with the number of issues. A general learning agent could help with these challenges by scaling the agent both in breadth (by handling multiple domains and opponents) and in depth (by handling large outcome spaces). Against this background, we propose a modular reinforcement learning (RL) framework to develop general negotiation agents. Reinforcement learning optimization methods have shown great learning ability in many different environments [19], against many different opponents [22, 23] and in very large state spaces [19, 22, 23].

Formulating a generic reinforcement learning framework for multi-issue negotiation is not a straightforward task. A major challenge is allowing for generic state and action spaces that can be represented in a compact manner and at the same time be defined arbitrarily by an agent designer to accommodate different negotiation settings. To facilitate the description of the states and actions we distinguish three major components involved in the strategy of the negotiating agent [3]: a bidding strategy which is used to decide what to offer, an opponent model that estimates the opponent's utility of different offers, and an acceptance strategy that is used to decide whether to propose the next bid or to accept that of the opponent. Through these components, the specifics of negotiation can be abstracted away by providing an interface for any reinforcement learning algorithm in order to learn a negotiation strategy.

The contributions of this paper to the existing body of work in this domain are threefold:

- We provide the RLBOA-framework: a universal Reinforcement Learning interface wherein non-repeated multi-issue alternating offers negotiations can be applied in a scalable way;

*These authors contributed equally to this paper.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

- We show a proof of concept by creating an agent from the framework. We show that the resulting RLBOA-agent can indeed learn generic negotiation strategies that are proficient across domains as well as opponents;
- Finally, the RLBOA-framework is implemented in the open-source negotiation software Genius [15], aiding further research into the topic.

2 RELATED WORK

This work is at the intersection of the domains of autonomous negotiation and reinforcement learning. In this section we discuss work done in both fields separately and combined.

2.1 Autonomous Negotiation

Research into autonomous negotiation can be roughly classified into two categories [17]. Theoretical approaches take methods from economics and game theory to classify distinct forms of negotiation and come to conclusions about optimal strategies and protocols. Computational approaches, on the other hand, mainly focus on developing and evaluating strategies in realistic tasks. Different computational methods have been used to increase the performance of autonomously negotiating agents and their decision making model; for example, Bayesian learning [13, 31], utility-graphs for non-linear preferences [20], Q-learning [7, 8, 28], and evolutionary computation [18]. In this paper we will focus on the agent decision making model, or negotiation strategy.

The design of a negotiation agent can be systematically analyzed by its components in the *BOA* framework [3]. The BOA framework decouples the complete negotiation strategy into a *B*idding strategy that decides what to offer, an *O*pponent model that models the opponent's preferences by estimating the utility of different offers, and an *A*cceptance strategy which is used to decide whether to propose the next bid, to accept the opponent's bid, or to end the negotiation. Later formulations of the BOA framework also define an opponent model strategy that specifies how the bidding strategy uses the information in the opponent model [12].

2.2 Reinforcement Learning

Reinforcement learning (RL) is a goal oriented optimization technique that learns a mapping from states to actions, called a *policy*, to control the behavior of an agent [25]. This is achieved by trial-and-error during repeated interaction with an environment. The environment is everything that is out of the immediate control of the agent [25]. Formally, the environment is defined as a Markov Decision Process (MDP) consisting of a set of states, S, their associated actions A_s, transition probabilities $P(s_{t+1}|s_t, a_t)$, rewards $r(s_t, a_t, s_{t+1})$ and a discount factor γ. The transition probability is the probability of arriving in state s_{t+1} given the current state s_t and the performed action a_t. The reward is a signal from the environment signifying the value of performing a specific action in a state. In practice however, the environment does not always fit such a description [19]. Although typical theoretical guarantees on convergence no longer hold in such cases [25], studies still find convergence empirically under various conditions [6, 26, 28].

RL has shown great promise in complex problems. For example, Tesauro's TD-Gammon learned to play the game of Backgammon [26]. Atari video games, which have an infinite state space, were learned to be played on a super-human level [19]. In addition, the World Champion in the game of Go was beaten by an RL agent decades before it was conceived to be possible for an AI system to accomplish such a feat [22, 23]. Finally, RL agents have the ability to gain a real understanding of the games they play and come up with creative new strategies and insights [23].

2.3 RL in Autonomous Negotiation Domains

Several studies have looked at the application of RL to automated negotiation. However, either the used methods or the focus within the automated negotiation domain always differed from ours. Tessauro and Kephart [28] examined three different oligopolistic economical settings with two competing sellers that do not have a deadline. In each setting, collusion would be the optimal strategy so as to avoid a price war. They found that against a deterministic opponent, their Q-learning approach was always able to find the optimal behavior. However, they also found that against opponents that are adaptive, the optimal strategy is much harder to learn since it is non-stationary and becomes history dependent. Further work also looked at the effects of different function approximators in the same scenario and found that regression trees were superior to neural networks, but that Q-tables worked better than both function approximators for small domains [24, 27]. In our implementation, we also use Q-learning as an optimization method, but where they focus on just three different domains, we train and evaluate the RLBOA-agent on a broader range of scenarios. Moreover, their agents, as opposed to the agents in our setting, have near perfect information. This makes it a two-player perfect-information deterministic game, a game very similar in spirit to chess.

Another important consideration is the work of Lewis et al. [14] who showed that applying reinforcement learning could improve the performance of a pre-trained Neural Network that was trained on human-human negotiations. They used a multi-issue bargaining set-up, but their focus was on the natural language side of negotiation, which can be used to increase the applicability of human-machine negotiation. Rodriguez-Fernandez et al. demonstrated the effectiveness of a Q-learning strategy for a decision support system for the Energy Market [21]. Their agent was able to assist in determining the context of the negotiation during the pre-negotiation phase. Where their work focuses on decision support systems, our research is in the field of completely autonomous negotiation.

3 NEGOTIATION MODEL

The negotiation proceeds through the stacked alternating offers protocol, in which the negotiation consists of rounds of consecutive turns. Each turn one party can either make an offer, accept, or leave the negotiation [1]. It is not possible for agents to recall an outstanding offer. The negotiation ends if the parties find a joint agreement or a deadline is reached. The deadline is measured in the number of rounds t. Negotiations are non-repeated, meaning that the actions an agent takes during one negotiation session will not impact actions the opponent takes in subsequent sessions.

The issues are the objects of the negotiation and, depending on their number, define either a single- or multi-issue negotiation. A possible outcome is an assignment of an option for each issue.

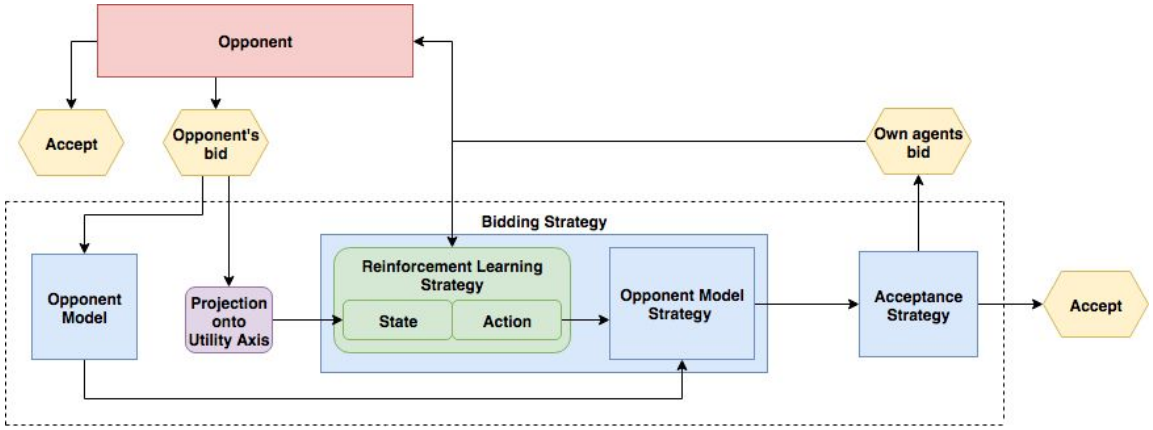


Figure 1: A schematic overview of the RLBOA-framework (within the dashed box), the BOA framework (blue) with the integrated RL component (green), the mapping onto the utility axis (purple), and the greater negotiation setting including the opponent (red) and negotiation objects (yellow). The arrows represent the information flow within the framework.

The Cartesian product of negotiation issues is called the *outcome space* and is formally denoted as $\Omega = \{\omega_1, \ldots, \omega_N\}$ where ω_i is a possible outcome and N is the total number of possible outcomes. For example, consider a buyer and a seller of a customizable laptop, where the issues are the components of the laptop (e.g., RAM, CPU, GPU, screen size, etc.) and the items are different options for each issue (e.g., a 13, 15 or 17 inch screen). The outcome space consists of all possible laptop configurations.

Each agent assigns a utility to an outcome ω, based on their preference profile, denoted by $U_A(\omega) \in [0, 1]$ for agent A. The outcome space in combination with the preference profiles defines a negotiation scenario. Both agents solely know their own preference profile and can only try to infer the preferences of the opponent by interpreting their bids. Both agents have a reservation value $r \in [0, 1]$, which is the minimal utility the agents find acceptable. An agent will not accept offers that do not meet this value.

4 RLBOA

In this section we provide an exposition of the proposed *RLBOA-framework*, a modular framework that facilitates the creation of autonomous negotiation agents using *reinforcement learning* (RL). We first introduce the components of our framework and then provide an instantiation of an RLBOA agent that exemplifies the framework. The effectiveness of this agent is subsequently demonstrated empirically.

4.1 The RLBOA-Framework

In order to provide a reinforcement learning interface for negotiation, we need to describe the states and actions that facilitate interaction with the environment in a domain-independent way. This is made difficult by two characteristics of the outcome space: first, its size is exponential in the number of issues, and second, the outcomes are unique for every combination of issues; e.g. outcomes in the energy domain cannot be directly compared with outcomes in a negotiation on a holiday destination. Our framework aims to address both these problems.

The RLBOA-framework allows for a compression of the state- and action space that facilitates the optimization of the bidding strategy by any reinforcement learning algorithm in such a way that it can be applied to different negotiation settings. The main characteristic of the framework is its modularity, allowing the agent designer to select a combination of components that is suitable for the task at hand. To this end we build on the existing BOA-framework [3] which allows a decoupling of the negotiation strategy into a bidding strategy, an opponent model, and an acceptance condition. In principle, each of these could be optimized with RL, but we choose to focus in the remainder on optimizing the bidding strategy only. We extend the existing modularity of BOA by allowing the use of any RL algorithm as well as any opponent model or acceptance strategy. This makes it an easy to use and broadly applicable framework. The fact that the same agent can learn from different settings is of great value for the creation of cross-domain negotiators. Figure 1 sketches an overview of the framework; the reminder of this section details the individual components.

4.1.1 Compressing the Outcome Space. Since the state- and action space as well as the diversity in negotiation settings are very large, we propose a reduction of state- and action space by representing every outcome ω_i by its corresponding utility for the RLBOA agent $U_A(\omega_i)$. We believe that this is a sufficient mapping that ensures two things. First, it ensures that the state space is always numerical. This allows for the use of any reinforcement learning method as an optimization technique. Second, it makes the framework agnostic to different negotiation scenarios since all outcome spaces in our negotiation model can be mapped onto the utility axis (see also Section 4.2.1).

4.1.2 Modular Strategy Components. The decoupled nature of BOA agents allows all of their components to be implemented independently. This permits us to only learn the bidding strategy and choose off the shelf components for the opponent model and acceptance strategy. This reduces the complexity of the problem by eliminating the need to simultaneously also learn the opponent's preferences, or whether or not to accept an offer. Moreover, the

BOA framework provides a natural way to translate the result of actions taken in the utility space back to the outcome space (Figure 1). Actions result in a target utility $\tilde{u}_A$. The opponent model strategy uses this target in combination with the current estimate of the opponent model to produce a candidate offer $\tilde{\omega}$. Finally, the acceptance condition determines whether to communicate the offer or settle for an agreement. It is important to note that every component outside of the RL module is considered as part of the environment, from the RL agent's perspective.

4.1.3 Reward Function and Training Pipeline. The framework gives the user the flexibility to choose any combination of (numerical) values from the negotiation model as a representation of states and actions. For example, a very simple state representation that uses the utilities of the opponent's last bid and your own last bid is enough. One can add any statistic derived from the negotiation environment, like distance to the Pareto frontier (if available) or the opponent's concession rate. It is also possible to add a time component. The action description can then be a target utility which will be passed on to the opponent model strategy (Figure 1). Moreover, any reinforcement learning algorithm can be used to train the agent based on a custom reward function. A natural reward signal could be the utility of an agreement, but other measures can be used based on the goal of the agent (e.g. utility functions based on social welfare or time). When using off-policy RL methods, it is sufficient to just have training data to train an agent. A simulation environment or even a complete description of the MDP is not needed. This can be useful when the agent needs to be trained for real-life applications, where simulation or a complete description of the MDP may not be available, but raw data is.

4.2 Using The Framework - RLBOA-Agent

Our RLBOA-framework can be instantiated by defining it's components (indicated in Figure 1). In our implementation we use of the shelf components for the opponent model and the acceptance strategy. We focus on showing how the RL-Strategy can be integrated.

4.2.1 State- and Action Space. The outcome space is discretized into evenly spaced utility bins, based on the utility function of the RLBOA agent (Figure 2). A bin is defined as

$$ub(\omega_A^t) = \lfloor U_A(\omega_A^t) \times N_{bins} \rfloor$$
$$ub(\omega_B^t) = \lfloor U_A(\omega_B^t) \times N_{bins} \rfloor$$

where $U_A(\cdot)$ is the utility function for agent A and N_{bins} the number of bins. A state s_t can then be represented as

$$s_t = \{ub(\omega_A^t), ub(\omega_B^t), ub(\omega_A^{t-1}), ub(\omega_B^{t-1}), t\}$$

where $ub(\omega_A)$ is the utility bin of the RLBOA agent for its own bid, while $ub(\omega_B)$ is the utility bin of the RLBOA-agent for the opponent's bid. The time is normalized between 0 and 1 as the percentage of progress made towards the deadline. This value is discretized in the same way as the utility. Both the time bins and the utility bins are evenly spaced between their minimum and maximum values. Figure 2 provides an example of how a utility bin is represented. Note that our state representation is thus composed of four utility bins and a time bin; the figure shows an example of one of the utility bins. When no bids have been placed yet, the missing bins are represented by a special token. This specific state-space description is in some sense a minimal working example that still allows to learn behavior that depends on the immediate bidding history (last round), as well as on the urgency of completing the negotiation before the deadline.

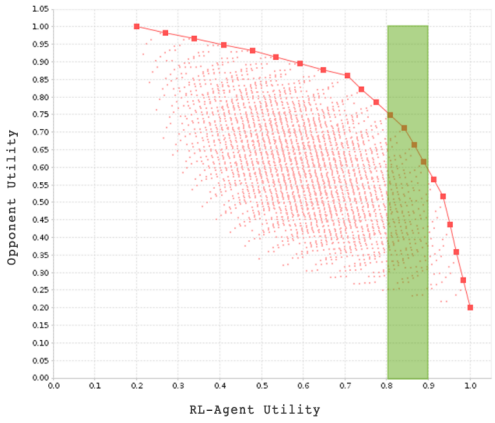


Figure 2: An example of a utility bin (green bar) in the outcome space. All pink dots are possible outcomes and the red curve is the Pareto frontier.

An action consists of either going up one bin (retracting offer), going down one bin (conceding offer), or staying in the same utility bin (idle offer). There is a special case for the opening offer. In this case the action space consists of the entire range of bins. There are two ways in which the RLBOA-agent can propose an action that results in selecting an empty bin. If the agent exceeds the limits of the outcome space, determined by the lowest acceptable utility $\max(ub^A_{min}, ub^A_{res})$, where ub^A_{res} is the reservation price bin, and the highest obtainable utility ub^A_{max}, the environment responds by bouncing back, meaning that it stays in the current bin. In other cases, when an empty bin is selected the environment skips over that bin in the direction of the proposed action. Once a target bin is selected by the RLBOA-agent, the opponent model strategy picks a bid based on the estimated preferences of the opponent. Specifically, the opponent model strategy proposes the offer that has the highest expected utility for the opponent in that bin.

4.2.2 Optimization Method. To optimize the RLBOA-agent's strategy we use Q-learning [30]. This is a model-free RL algorithm that iteratively improves an estimate of the state-action value function by bootstrapping on previous estimates while sampling state transitions from the environment. In particular, the agent samples tuples of the form $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ from experience and then updates the estimated state-action value function $Q(s, a)$ as follows:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \cdot \max_{a'} Q(s_{t+1}, a') \right].$$

The Q-learning agent selects actions by sampling from a behavior policy π that is ϵ-greedy with respect to the Q-function, i.e., $a_t = \arg\max_a Q(s_t, a)$ with probability $(1-\epsilon)$, and with probability ϵ a random action is taken in order to balance exploration and exploitation [25]. After a policy has been learned, exploration is disabled by setting $\epsilon = 0$ during evaluation.

In our negotiation domain, rewards are zero in all states except for the terminal state, where the reward equals the utility of the final agreement, or the reservation value $r \in [0, 1]$ when no agreement is reached before the deadline.

5 EXPERIMENTAL SETUP

We conduct two experiments in order to demonstrate the learning capabilities of our RLBOA-agent. The goal of the first experiment is to show that the agent is indeed capable of learning a meaningful policy across domains where the outcome space differs in size and (types of) issues. The second experiment tests whether the same agent can also learn in an environment where its opponent's strategy varies across negotiations. Rather than generating a state-of-the-art autonomous negotiating agent, our purpose is to demonstrate the feasibility of our reinforcement learning framework for developing autonomous negotiating agents. We analyze the agent in six different *scenarios* against a pool of four different *opponents*, which will be discussed later in this section.

Opponent Model. A simple frequency model is used as opponent model of the agent. This model keeps track of a ranking of issues based on how many times a certain item appears in the opponent's bidding history [29]. Even though more accurate Bayesian opponent models exist [3], we opted for a computationally less expensive one. The benefits in terms of utility of more complex models are usually not significant [29].

Acceptance Condition. Acceptance strategies can be seen as utility based, time-based or a combination of the two [4]. Our agent uses a utility-based acceptance strategy, $\mathbf{AC}_{next}(\alpha, \beta)$:

$$A^a(t^t, \omega^t_{B\to A}) = \begin{cases} End & \text{if } t > T, \\ Accept & \text{if } \alpha \cdot U_A(\omega^t_{B\to A}) + \beta \geq U_A(\omega^{t+1}_{A\to B}), \\ \omega^{t+1}_{A\to B} & \text{otherwise.} \end{cases}$$

where $\omega^t_{B\to A}$ is the bid at time t from the opponent B to agent A, T is the (universal) deadline, α is a scaling factor, β is the minimally acceptable 'utility gap' agent A allows, $\omega^{t+1}_{A\to B}$ is the counter-offer agent A would send at time $t+1$ to the opponent and $U_A(\omega)$ is the utility the agent obtains from a bid. Specifically, we set $\alpha = 1$ and $\beta = 0$. Conceptually, the agent accepts an offer if its utility exceeds that of the agents counter offer.

Hyperparameters. Several hyperparameters influence the performance of the RLBOA-agent, which we tune using a small hyperparameter search. The objectives to which the hyperparameters are tuned are the number of episodes to convergence and the utility obtained at the stable policy.

The effect of an increase in the number of utility bins N_{ub} is twofold. Firstly, it influences the impact of a concession. More bins means a smaller concession or retraction at each turn. Secondly, the state space is exponential in N_{ub}. Experiments varying the value of this parameter showed that for larger values it took the agent longer to converge on a policy. We set $N_{ub} = 10$ for all experiments, which we found to be a good balance between convergence time and performance. The number of time bins N_{tb} is set to 5, allowing the agent to differentiate between two early stages, a middle stage and two final stages of the negotiation session.

We use an ϵ-greedy exploration policy that picks $\arg\max_a Q(s, a)$ with probability $1 - \epsilon$ and a random action otherwise. The random factor in the policy ensures all states will be seen, given enough time. The best performing exploration rate was found to be $\epsilon = 0.1$, which we subsequently use in all experiments. The learning rate α is set to 0.15. The learning process behaves differently based on the initial values in Q. For example, an agent designer may want an agent to start with a bid that is high enough to leave enough room to concede. We limit the use of such heuristics in our reinforcement learning agent to show that our method is able to learn these strategies on its own.

5.1 Negotiation Settings

Our first set of experiments are *scenario generality* experiments, in which the scenarios that the RLBOA-agent encounters during training are randomly selected for each negotiation, while the opponent remains fixed. We run this experiment four times, once for each opponent. The policy that the agent learns in each experiment is evaluated on the same individual negotiation settings it encountered during training (i.e. against the same opponent and all scenarios).

In the second set of experiments, the *opponent generality* experiments, the scenario is kept fixed while the opponents are randomized during training. Evaluation is then performed in the same scenario against all opponents. In all experiments the agents are trained until convergence.

Scenarios. We consider six different scenarios. They vary in size of the outcome space $|\Omega|$ and *opposition*. Opposition is a standard metric in the yearly Autonomous Negotiation Agent Competition [2] and is defined as the minimal euclidean distance to the optimal, but not necessarily achievable, outcome for both parties $\hat{\omega}$:

$$\text{opposition}\,(\Omega) = \min_{\omega\in\Omega} d(\omega, \hat{\omega})$$

In our case $\hat{\omega} = \omega$ s.t. $(U_A(\omega), U_B(\omega)) = (1, 1)$. It is an indication of how aligned the interests of the agents are. This is important for the resulting strategy, e.g. a cooperative strategy might work better in low opposition domains and a non-cooperative strategy might work better in high opposition domains. For reference, in a zero-sum game for two players, $opposition \approx 0.71$. We created six scenarios: a small, a medium and large domain in terms of outcome space and for all three domains one scenario with low and one scenario with high opposition.

In Table 1 an overview can be found for the characteristics of these scenarios. In every other aspect, the scenarios are equivalent. These characteristics were chosen to explore the effects of size and opposition on learning in the RLBOA-framework. Robustness to size is an explicit goal of the framework and therefore of key interest. Opposition is an important factor in what makes an appropriate negotiation strategy. Low opposition domains leave more room for cooperation and win-win outcomes whereas high opposition domains are more like zero-sum games.

Opponents. To ensure a diverse pool of opponents, we follow the categorization of various families of strategies made by Faratin et al. [9]. Specifically we generate several instances of the time dependent agent family and the behavior dependent agent family.

Table 1: Overview of the scenarios. We consider three domains, each with two sets of preference profiles (low and high opposition).

Domain	Outcome space	Low opp.	High opp.
Small	256	0.2615	0.5178
Medium	3.125	0.3111	0.5444
Large	46.656	0.2595	0.5250

Agents of the *time dependent agent family* concede more rapidly when the deadline approaches. The agents within this family can be differentiated by the shape of the concession curve the agents exhibit (i.e. at what rate they concede over time). Candidate offers are generated as follows [11]:

$$\tilde{\omega}^{t+1}_{A\to B} = U^A_{min} + F^A(t)(U^A_{max} - U^A_{min})$$

where $\tilde{\omega}^{t+1}$ is the next candidate offer, U^A_{min} is the lowest obtainable utility above the reservation value, U^A_{max} is the highest obtainable utility for agent A and the function F^A is modelled as

$$F^A(t) = k_A + (1 - k_A)\left(\frac{\min(t, T)}{T}\right)^{\frac{1}{e}}$$

where $k_A \in [0, 1]$ is the utility of the first bid offered by the time dependent agent. At $k = 0$ the agent opens with its maximum utility bid; the parameter e then determines the manner in which the agent concedes and defines the shape of the concession curve. There are an infinite number of possible agents within this family, but two sets of agents show distinctive different behavior: *Boulware agents* with $e < 1$; and *conceder agents* with $e \geq 1$ [11]. Boulware agents concede very slowly until the deadline is almost reached and then concede to the reservation value very quickly. In contrast, conceder agents move towards their reservation value rather quickly, with $e = 1$ being a linear conceder. As our focus is on negotiation settings with a deadline, this is a reasonable family of agents to have in our opponent pool.

The second family of agents in our opponent pool is the *behavior dependent* family, often referred to as *Tit-for-Tat agents*. Agents of this family base their actions on the opponent's actions. Specifically, they imitate the opponent's behavior to a certain degree to avoid exploitation. Within this family Faratin et al. [9] make a distinction between *absolute imitation*, *relative imitation* and *average imitation* over the last few turns. The behavior dependent agent we use is inspired by the average Tit-for-Tat agent.

The average Tit-for-Tat agent uses a percentage change in a window of $\gamma \in [1, t]$ bids over the opponent's history to determine its bid [9]:

$$\tilde{\omega}^{t+1}_{A\to B} = \min\left(\max\left(\frac{u^{t_n-2\gamma}_{B\to A}}{u^{t_n}_{B\to A}} u^{t_n-1}_{A\to B}, U^A_{min}\right), U^A_{max}\right)$$

Here, $\tilde{\omega}^{t+1}$ is again the new candidate offer and the ratio is the relative change of utility of the opponent's offer over the past 2γ turns. This factor is applied to the utility value of the time-dependent agent's last offer to produce the next candidate offer, which is clipped to the range of the agent's reservation value and its maximum obtainable utility. Before the agent has enough information to utilize its imitative behavior it goes through the possible bids in descending order of its own utility.

Our final opponent pool consists of two time dependent agents, with $e \in \{0.1, 1.0\}$ and $k = 1.0$, and two behavior dependent agents, with $\gamma \in \{1, 2\}$. All agents are implemented as simple BOA agents with the strategies described above as their bidding strategy and the same acceptance strategy as the RLBOA-agent: $\mathbf{AC}_{next}(\alpha, \beta)$. These agents don't have a model of their opponent preferences, so the opponent modelling component is irrelevant.

Since the opponents are relatively simple, the optimal strategy to be used against them is easily identifiable. Against a time dependent opponent an agent can generally maximize its utility by consistently offering their maximum utility outcome and wait for its opponent to concede all the way to an agreement. Against behavior dependent opponents it is optimal to produce a sequence of offers that is increasing in the utility of the opponent, while you concede as little as possible in terms of your own utility. The perceived concession triggers the opponent to reciprocate. Knowing these optimal policies allows for a qualitative comparison of the learned strategy of the RLBOA-agent.

5.2 Evaluation Metrics

The performance of the RLBOA-agent is measured as the utility obtained by following a strictly greedy policy on a converged Q-table. Specifically, we compare this utility to the RandomAgent. This is a BOA agent with the same components as the RLBOA-agent, except for the use of a bidding strategy that simply picks random offers. In settings where most offers are valued very highly, the RandomAgent will obtain a high utility agreement often with low variance. In settings were the utility of bids is more spread out, this will be expressed in the mean and variance of utilities obtained by repeated negotiations by the RandomAgent. Therefore the mean and variance are good measures of how much of the obtained utility is due to the distribution of outcomes.

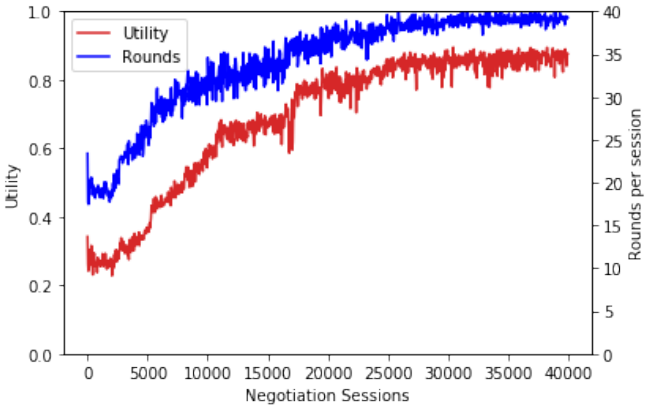
To obtain baselines for the mean utility and its variance in a given setting we aggregate the result of 100 negotiations by the RandomAgent. Obtaining a utility that is reasonably higher than the expectation under the RandomAgent is an indication whether the RLBOA-agent has developed a good strategy. We define reasonably higher as being more than two standard deviations from the mean. Due to the low number of samples we cannot claim statistical significance. However, it does show that the learned behavior results in a utility that is well above the expected value of the negotiation setting, which demonstrates the capabilities of the RLBOA-agent.

6 RESULTS

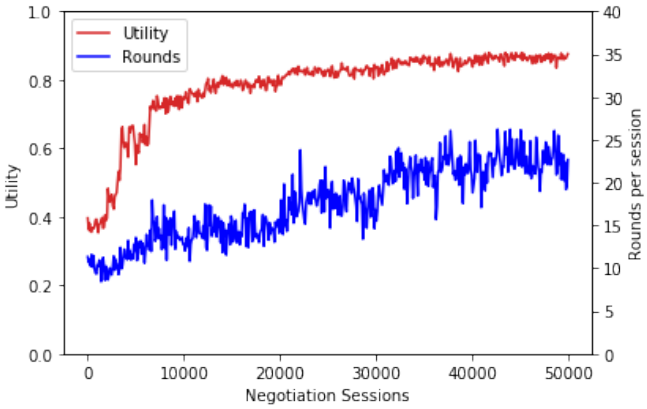
In the following section the results of the experiments from the previous section are presented. First, we look at the quantitative results of both the scenario and domain generality experiments. Next, we will discuss some qualitative results in the form of descriptions of commonly observed agent behavior and a closer look at the individual negotiation settings.

6.1 Scenario Generality

It is immediately clear from Figure 3a that the RLBOA-agent steadily improves its return during training. In this specific experiment,



(a) Scenario generality experiment against the Boulware agent.



(b) Opponent generality experiment in the medium sized domain with low opposition.

Figure 3: Learning curve of RLBOA-agents. The curves show the progress in utility and rounds played over sessions.

against the Boulware agent, the policy converges after approximately 40.000 negotiations. The learned policy at that point evaluates at an average utility of 0.88 over all domains, which is 0.41 higher than the RandomAgent. The duration of a negotiation session is highly correlated with the utility that is obtained in negotiations against time dependent agents. This is not surprising, as these agents are designed to offer more attractive bids over time. What the RLBOA-agent learns is to hold off an agreement for as long as it can, which is optimal against time dependent agents.

Looking at the utilities in the individual settings in Table 2a, it can be seen that all four policies perform much better than random across scenarios. This means that the RLBOA-agent is able to learn a policy that is agnostic with respect to to the specifics of the domains. This result is promising for using the RLBOA-framework for creating agents that negotiate in more realistic settings. This result is strongest against time dependent agents, where the policy performs two standard deviations above the RandomAgent in every evaluated setting. Overall the results in Table 2a show the RLBOA-agent learned how to negotiate well in 41 of the 48 negotiation settings. In 23 of the 48 settings it obtained a utility above 0.90, which means that the RL-strategy could not have performed better, as the agreement ended up in the highest utility bin. Moreover, all the outcomes in each scenario are approximately normally distributed across the utility axis, with a mean around 0.6 (see Figure 2 for an example). Therefore very few outcomes are expected in the $[0.9, 1]$ utility range, and thus achieving such an outcome is a strong indicator for a successful negotiation policy.

6.2 Opponent Generality

From the results in Table 2b it is clear that the RLBOA-agent has learned a non-trivial policy in all experiments. The agent learned strategies that performed well in in 39 of the 48 negotiation settings, while in 14 out of the 48 settings it obtained a utility score higher than 0.9. Similar learning curves to the scenario generality experiments are observed (see Figure 3b for an example). The ability to generalize, however, is less for opponents than that it is for scenarios. In two out of six experiments, the agent generalizes perfectly over opponents. On the other hand, there are also experiments in which the performance of the policy is skewed towards the time dependent agents. This suggests that there is a trade-off in performance against the two types of opponents, and the time dependent agents have higher value. This dynamic is discussed further in subsection 6.3, in the qualitative analysis. Convergence in these experiments is related to the size of the scenario. In small scenarios, the policy converges after around 20.000 negotiations. This number is 30.000 for medium and 35.000 for large scenarios.

6.3 Qualitative Analysis

To appreciate the results in full, it is worth looking at behavior of the RLBOA-agent qualitatively. We discuss some general observations below.

6.3.1 Scenario Generality. From inspecting the behavior of the RLBOA-agent in different scenarios, we can see that it consistently learns to open the negotiation with an offer in the highest utility bin. Against the behavior dependent agents it tries not to concede too much, while waiting for the opponent model to learn how to offer better bids to the opponent. Against the time dependent agents the RLBOA-agent also learned to hold out in the higher utility bins and wait for the time dependent agent to concede.

6.3.2 Opponent Generality. For the opponent generality experiments we observed that the RLBOA-agent generally opens in a high utility bin and then stays around that bin. As we observed before, this is a decent strategy against all agents. The only setting in which this does not seem to work well is against the relative Tit-for-Tat agent in the small domain with high opposition. Intuitively, this might be explained by the fact that against the relative Tit-for-Tat agent the RLBOA-agent must offer the opponent consecutively better bids to make it concede. However, in the small domain with high opposition there is little room to offer a better bid within a given utility bin, and therefore the RLBOA-agent needs to concede in order to obtain an agreement.

Table 2: Results of scenario and opponent generality experiments. Utilities obtained by evaluating the learned policy in different negotiation settings. Left: RL-Agent was first mover, right: RL-Agent was second mover. Bold entries indicate that the utility was at least two standard deviations higher than the average utility of the RandomAgent in the same setting.

(a) Scenario generality. Each *row* represents one policy. The utilities are those obtained by that policy in the corresponding setting.

Opponent \ Scenario	*Small-Low*	*Small-High*	*Medium-Low*	*Medium-High*	*Large-Low*	*Large-High*
AverageTitForTat1	0.80; 0.60	0.71; 0.63	**0.94**; **0.82**	**0.73**; 0.63	**0.95**; 0.80	**0.81**; **0.73**
AverageTitForTat2	**0.94**; **0.81**	**0.73**; **0.64**	**0.94**; **0.82**	**0.72**; 0.67	**0.88**; 0.70	**0.76**; **0.81**
TimedependentLinear	**0.91**; **0.95**	**0.92**; **0.90**	**0.91**; **0.91**	**0.91**; **0.90**	**0.90**; **0.90**	**0.91**; **0.95**
TimedependentBoulware	**0.96**; **0.92**	**0.82**; **0.83**	**0.90**; **0.91**	**0.81**; **0.83**	**0.90**; **0.92**	**0.91**; **0.89**

(b) Opponent generality. Each *column* represents one policy. The utilities are those obtained by that policy in the corresponding setting.

Opponent \ Scenario	*Small-Low*	*Small-High*	*Medium-Low*	*Medium-High*	*Large-Low*	*Large-High*
AverageTitForTat1	0.81; 0.77	0.00; 0.00	**0.94**; **0.82**	0.00; 0.62	**0.89**; 0.81	**0.81**; **0.71**
AverageTitForTat2	**0.94**; **0.86**	**0.82**; 0.00	**0.94**; **0.82**	**0.72**; **0.70**	**0.89**; 0.81	**0.82**; **0.63**
TimedependentLinear	**0.91**; **0.86**	**0.92**; **0.90**	**0.91**; **0.91**	**0.84**; **0.84**	**0.86**; **0.90**	**0.84**; **0.86**
TimedependentBoulware	**0.83**; **0.86**	**0.91**; **0.92**	**0.90**; **0.91**	**0.81**; **0.90**	**0.81**; **0.81**	**0.70**; **0.89**

7 CONCLUSION AND DISCUSSION

We propose RLBOA, a modular framework that facilitates the creation of negotiating agents that use reinforcement learning to learn efficient policies that can generalize over a range of negotiation scenarios. The modularity of the RLBOA-framework allows agent designers to pick and choose from a wide range of components to design an agent that fits the task at hand. Our experimental results show that the RLBOA-agent can generalize over both different opponents and different scenarios. The RLBOA-agent learns to effectively exploit both time dependent and behavior dependent opponents by either making use of its opponent model or the opponent's tendency to concede over time.

While our experiments focus on a bilateral round-based alternating offers negotiation protocol, we believe that our formulation of negotiation as a reinforcement learning problem can be adapted easily to other forms of negotiation. For example, in a multi-agent setting, the state representation can be extended by adding utility bins for each opponent. Moreover, if the deadline is measured in wall-time, the agent can seamlessly incorporate such a time factor in its state description. The rest of this section outlines some of the possible avenues of future research.

With respect to our experimental setup, it is clear that our opponent pool and the used set of scenarios are not exhaustive. Furthermore, all learning was done in absence of a discount rate, which affects the strategies due to a lack of incentive for the agent to close deals faster. Further research is needed to examine the behavior of the RLBOA-agent in a broader set of negotiation settings. Another important realization is that, as our approach utilizes the BOA-framework, the performance of the RLBOA-agent depends on the performance of other components besides the bidding strategy. Continuation of this research could test the RL-strategy with different combinations of components. At the same time, extensions to the RLBOA-framework are possible where other BOA components are optimized through RL. An extension where the acceptance strategy is optimized can be implemented relatively easily by using the same mapping of outcomes onto utility, together with any RL-algorithm. For the opponent model, the RLBOA-framework cannot be extended as naturally, since the opponent model does not perform actions. Other interesting extensions could be a new meta-component that learns to select an (RL-)strategy from a library of strategies. Also, the framework is dependent on a (possibly estimated) utility. In real-world settings, the agent must thus also be capable of obtaining these utilities from the end-user.

Finally, a logical approach for extending the capabilities of the model is the application of function approximators (such as deep neural networks) and other advanced reinforcement learning techniques that improve sample efficiency (such as eligibility traces). While function approximators are more complex and less stable to train, they can be used to to learn a better state representation given all the available information about the negotiation. Examples of this are an efficient representation of the bidding history or opponent strategies on a meta-level, which might increase the performance in terms of utility and generality.

ACKNOWLEDGMENTS

This work is part of the Veni research programme with project number 639.021.751, which is financed by the Netherlands Organisation for Scientific Research (NWO). This project has received funding in the framework of the joint programming initiative ERA-Net Smart Energy Systems' focus initiative Smart Grids Plus, with support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 646039. We are indebted to the anonymous reviewers of AAMAS 2019 for their valuable feedback.

REFERENCES

[1] Reyhan Aydoğan, David Festen, Koen V Hindriks, and Catholijn M Jonker. 2017. Alternating offers protocols for multilateral negotiation. In *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 153–167.
[2] Tim Baarslag, Katsuhide Fujita, Enrico H Gerding, Koen Hindriks, Takayuki Ito, Nicholas R Jennings, Catholijn Jonker, Sarit Kraus, Raz Lin, Valentin Robu, et al.

2013. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. *Artificial Intelligence* 198 (2013), 73–103.
[3] Tim Baarslag, Koen Hindriks, Mark Hendrikx, Alexander Dirkzwager, and Catholijn Jonker. 2014. Decoupling negotiating agents to explore the space of negotiation strategies. In *Novel Insights in Agent-based Complex Automated Negotiation*. Springer, 61–83.
[4] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. 2013. Acceptance conditions in automated negotiation. In *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Springer, 95–111.
[5] Tim Baarslag, Michael Kaisers, Enrico Gerding, Catholijn M Jonker, and Jonathan Gratch. 2017. When will negotiation agents be able to represent us? The challenges and opportunities for autonomous negotiators. (2017).
[6] Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. 2009. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*. 1204–1212.
[7] Henrique Lopes Cardoso and Eugenio Oliveira. 2000. Using and evaluating adaptive agents for electronic commerce negotiation. In *Advances in Artificial Intelligence*. Springer, 96–105.
[8] Lihong Chen, Hongbin Dong, Qilong Han, and Guangzhe Cui. 2013. Bilateral multi-issue parallel negotiation model based on reinforcement learning. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 40–48.
[9] Peyman Faratin, Carles Sierra, and Nick R. Jennings. 1998. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems* 24, 3-4 (1998), 159–182.
[10] Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. 2014. *Principles of automated negotiation*. Cambridge University Press.
[11] S Shaheen Fatima, Michael Wooldridge, and Nicholas R Jennings. 2001. Optimal negotiation strategies for agents with incomplete information. In *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 377–392.
[12] MJC Hendrikx. 2012. Evaluating the Quality of Opponent Models in Automated Bilateral Negotiations. (2012).
[13] Koen Hindriks and Dmytro Tykhonov. 2008. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 331–338.
[14] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or No Deal? End-to-End Learning for Negotiation Dialogues. *ArXiv e-prints* (2017). arXiv:cs.AI/1706.05125
[15] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M Jonker. 2014. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence* 30, 1 (2014), 48–70.
[16] Raz Lin, Sarit Kraus, Jonathan Wilkenfeld, and James Barry. 2008. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence* 172, 6 (2008), 823.
[17] Fernando Lopes, Michael Wooldridge, and Augusto Q Novais. 2008. Negotiation among autonomous computational agents: principles, analysis and challenges. *Artificial Intelligence Review* 29, 1 (2008), 1–44.
[18] Noyda Matos, Carles Sierra, and Nicholas R Jennings. 1998. Determining successful negotiation strategies: An evolutionary approach. In *Multi Agent Systems, 1998. Proceedings. International Conference on*. IEEE, 182–189.
[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[20] Valentin Robu, DJA Somefun, and Johannes A La Poutré. 2005. Modeling complex multi-issue negotiations using utility graphs. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 280–287.
[21] J Rodriguez-Fernandez, T Pinto, F Silva, I Praça, Z Vale, and JM Corchado. 2019. Context aware q-learning-based model for decision support in the negotiation of energy contracts. *International Journal of Electrical Power & Energy Systems* 104 (2019), 489–501.
[22] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.
[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354.
[24] Manu Sridharan and Gerald Tesauro. 2002. Multi-agent Q-learning and regression trees for automated pricing decisions. In *Game Theory and Decision Theory in Agent-Based Systems*. Springer, 217–234.
[25] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press, Cambridge, MA.
[26] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.
[27] Gerald Tesauro. 2000. Pricing in agent economies using neural networks and multi-agent Q-learning. In *Sequence learning*. Springer, 288–307.
[28] Gerald Tesauro and Jeffrey O Kephart. 2002. Pricing in agent economies using multi-agent Q-learning. *Autonomous Agents and Multi-Agent Systems* 5, 3 (2002), 289–304.
[29] Okan Tunalı, Reyhan Aydoğan, and Victor Sanchez-Anguix. 2017. Rethinking frequency opponent modeling in automated negotiation. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 263–279.
[30] Christopher J.C.H. Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.
[31] Dajun Zeng and Katia Sycara. 1998. Bayesian learning in negotiation. *International Journal of Human-Computer Studies* 48, 1 (1998), 125–141.