Autonomous Agents and Multi-Agent Systems (2021) 35:27
https://doi.org/10.1007/s10458-021-09513-x

ANEGMA: an automated negotiation model for e-markets

Pallavi Bagga[1] · Nicola Paoletti[1] · Bedour Alrayes[1] · Kostas Stathis[1]

Accepted: 28 May 2021 / Published online: 7 June 2021

Abstract
We present a novel negotiation model that allows an agent to learn how to negotiate during concurrent bilateral negotiations in unknown and dynamic e-markets. The agent uses an actor-critic architecture with model-free reinforcement learning to learn a strategy expressed as a deep neural network. We pre-train the strategy by supervision from synthetic market data, thereby decreasing the exploration time required for learning during negotiation. As a result, we can build automated agents for concurrent negotiations that can adapt to different e-market settings without the need to be pre-programmed. Our experimental evaluation shows that our deep reinforcement learning based agents outperform two existing well-known negotiation strategies in one-to-many concurrent bilateral negotiations for a range of e-market settings.

Keywords Multi-agent negotiation · Deep reinforcement learning · Concurrent bilateral negotiation

1 Introduction

Negotiation is a process in which different parties exchange offers in order to mutually explore the likelihoods of reaching agreements [17]. As such a process can be time-consuming for humans, and consequently expensive [19], automating it saves both time and money. To address this problem, autonomous agent systems have been advocated as a key technology for automating human negotiations [23]. In automated negotiation of this type, humans state their goals and agents engage in strategic interactions with other agents to achieve them. This approach has many advantages due to the adaptive and multi-processing capabilities of autonomous agents. For example, in e-commerce marketplaces [21, 41],

✉ Pallavi Bagga
pallavi.bagga@rhul.ac.uk

Nicola Paoletti
nicola.paoletti@rhul.ac.uk

Bedour Alrayes
balrayes@ksu.edu.sa

Kostas Stathis
kostas.stathis@rhul.ac.uk

1 Department of Computer Science, Royal Holloway University of London, Egham, Surrey, UK

autonomous agents offer systematically all possible outcomes achieving better cost and time efficiency than a human agent [16, 26, 32–34].

In this paper, we are more interested in open and dynamic e-markets such as E-bay.[1] In such a type of e-markets, buyer agents are engaged in concurrent bilateral negotiations with unknown seller agents using strategies to anticipate the opponent's actions or intentions automatically, without the need to be pre-programmed. However, the design of autonomous agents capable of learning negotiation strategies from concurrent interactions with other agents is still an important open problem. Aspects of this problem have been studied previously, but mainly using heuristic strategies [3, 35, 39, 55] that, as such, do not support learning. From these, strategies that can adapt to changes in the environment exist, for example see [55]. Approaches like [40, 61] use learning based on Genetic Algorithms (GA), but require a huge number of trials before obtaining a good strategy, which makes them infeasible for online settings. Previous works on Reinforcement Learning (RL)-based negotiation typically employ Q-learning [7, 10, 12, 42, 49, 53], but this type of learning does not support continuous actions. This is an important limitation in our setting because we want our agent to learn how much to concede, e.g. on the price of an item for sale, which naturally leads to a continuous action space.

In addition and independently of the approach, numerous works in the domain of bilateral negotiation rely on the Alternating Offers protocol [45] as the negotiation mechanism. This, despite its simplicity, does not capture many realistic bargaining scenarios.

We propose, to the best of our knowledge, the first Deep Reinforcement Learning (DRL) approach for concurrent bilateral negotiations in open, dynamic and unknown e-market settings. In particular, we define a novel DRL-inspired agent negotiation model called *ANEGMA* which allows the buyer agent to develop an adaptive strategy to effectively use against its opponents (which use fixed-but-unknown strategies) during concurrent negotiations in an environment with incomplete information. We choose to work with deep neural networks which provide a rich class of strategy functions to capture the complex decision logic behind negotiation.

RL approaches take long to find an optimal policy completely from scratch [60], which might hinder the online performance of a buyer agent. To address this problem, we pre-train our negotiation strategies based on DRL using supervised learning (SL) from a set of training examples. To overcome the lack of real-world negotiation data for the initial training, we generate a synthetic dataset using the simulation environment in [1] and two well-known strategies for concurrent bilateral negotiation described in [2] and [55], respectively. We observe that negotiation strategies pre-trained against a particular opponent strategy quickly adapt to different opponent strategies during online RL.

The contribution of this paper is mainly threefold:

- We propose a novel agent model for concurrent bilateral negotiation based on DRL and SL.
- To implement our model, we extend the simulation environment of [1] to support agent learning during concurrent bilateral negotiation.
- We perform rigorous experimental evaluations demonstrating that our approach outperforms the current state-of-the-art in one-to-many concurrent bilateral negotiations. Also, the agents adopting our model can quickly adapt to a range of e-market settings.

[1] https://www.ebay.com/.

The remainder of this paper is structured as follows. In Sect. 2, we discuss previous work related to learning for automated negotiation. Following the limitations of related work, in Sect. 3, we propose a DRL-based model for agent negotiation that addresses these limitations. Then, in Sect. 4, we describe how we generate synthetic supervision data, we identify performance measures, and describe the employed SL and DRL models accompanied with their corresponding algorithms. We, then, experimentally evaluate our proposed work by analysing the results we obtain using our model in Sect. 5. Finally, Sect. 6 summarises our conclusions and identifies directions for future work.

2 Related work

Developing models that let an agent learn a strategy during negotiation normally assumes the following three-phase process [25]: Phase-I (or the *pre-negotiation phase*) prepares the negotiating agent with details such as the settings and protocol, the negotiation parameters and number of issues, and a user preference model. It also possibly involves learning the user model from given partial information as well as eliciting the preferences from the user under uncertainty. In Phase-II (or the *negotiation phase*), the agent is deployed to negotiate, involving offer generation and additional components such as opponent model prediction, offer evaluation and acceptance. Finally, in Phase-III (or the *post-negotiation phase*), the optimality of the final agreement is assessed in terms of various metrics such as average individual or social welfare utilities, and distance to Nash equilibrium, only if an agreement is reached. In this work, we focus on Phase-II in order to learn a negotiation strategy. As a result, we compare our work with the extant literature that significantly addresses the *negotiation* phase. Also, we acknowledge that our approach differs from existing work in one or more of the following aspects: the way and the type of learning used, the focus and goal of the research, and the application domain.

Most of the work on learning in negotiation has focused significantly on learning the characteristics of the opponent agents. Hindriks et al. [20], Buffett and Spencer [9], Yu et al. [56], Zhang et al. [58], Li and Cao [29], and Zeng and Sycara [52, 57] propose methods that build on Bayesian learning to estimate the opponent's preferences. A similar method is used by Ren and Anumba [44], and Zhang et al. [59] to learn the opponent's acceptance strategy by estimating the reservation value from previous negotiations. Narayanan and Jennings [38] also learn an opponent's strategy by relying on Markov chains and Bayesian learning for single-issue negotiations. Bala and Mukhopadhyay [8] predicts the opponent's behaviour using artificial neural networks (ANNs) trained with data collected from past negotiations In contrast, our work abstracts away from learning an explicit opponent model, in that the opponent behaviour is implicitly represented within the agent's neural-network-based strategy. This strategy based on DRL can adapt to the behaviour of unseen opponent agents.

Another area of learning effort is on finding optimal offers. In this line of work, Lau et al. [27] employ an evolutionary learning approach using GA to find Pareto-optimal offers that can be mutually acceptable by the involved negotiating parties. Likewise, Choi et al. [13] in their work make an attempt to use GA to learn the opponent preferences based upon stochastic approximation in electronic business. Unlike other GA negotiation strategies, Choi et al. change the mutation rate dynamically to adapt to the environment. Choudhary and Bhardwaj [14] also examine GA-based learning techniques for multi-agent negotiation but the application domain focuses on group recommender systems, rather than negotiation. Moreover, Sim et al. [47, 48] propose a method which uses both Bayesian learning and GA for estimating the opponent's reservation price and generating a proposal during each negotiation round, respectively. In their approach, the opponent's decision function is assumed to have a particular form which is also used to directly estimate deadline of an opponent from the estimated reservation value. As we are interested in making offers using online learning, we rely on RL to obtain an optimal policy, as GAs may require a large number of iterations to achieve optimality, which can be time-consuming.

Moreover, much recent work has focused on applying RL in agent negotiation. Q-learning is used in [54] in single- and multi-agent settings for making economic decisions, in [42] for agent-human multi-issue negotiations, and in [10] to dynamically adapt the negotiation tactic. Building on [4], Bakker et al. [7] propose a modular RL-based BOA (Bidding strategy, Opponent model and Acceptance condition) framework. This framework uses tabular Q-learning to learn the bidding strategy, which entails discretizing the continuous state/action space. This is not, however, an ideal solution for large state/action spaces as it may lead to the curse of dimensionality, as well as cause the loss of relevant information about the state/action domain structure. Razeghi et al. [43] use Deep Q Networks [36] to design a learnable acceptance strategy based on feedback received from the environment. The main difference of the above Q-learning approaches, when compared to ours, is that they cannot be used in continuous action spaces [31], and thus are inappropriate for our setting.

There are also a number of related works where RL is integrated with other approaches. These include: [61] which combines evolutionary algorithms and reinforcement learning, [49] which combines regression trees with single-agent and multi-agent Q-learning, and [50] which proposes a bilateral price negotiation strategy based on Bayesian classification (for opponent modelling) and Q-learning (for generating a counter-offer). Previous work has also demonstrated the advantage of pre-training DRL agents using SL. Examples include the work of Lewis et al. [28], which is akin to ours in that they also combine SL and RL, but their focus is on natural language processing rather than autonomy in negotiations. Our more recent work [6] also takes a similar approach and learns strategies according to tactics, but does not consider concurrent negotiations, only single bilateral ones.

This paper is based on our previous work [5], but introduces a number of significant extensions as follows. Firstly, we include details of the concurrent negotiation protocol and background of the learning approaches used in the proposed model. Secondly, we explain how we extended the RECON simulation environment [1] to support strategies that are capable of online learning. Thirdly, we include a detailed analysis of related work in RL-based approaches for automated negotiation. Finally, we provide a more comprehensive experimental evaluation, including new results that did not appear in [5].

A comparison of RL-based approaches for automated negotiation is available in Table 1. We classify the relevant literature using the following attributes: *Continuous Action space*, whether or not the offer space is continuous or discretized (we put an hypen where we could not evince this from the paper alone); *concurrent negotiations*, whether or not agents negotiate concurrently with multiple other agents; *dynamic environment*, whether or not the environment can change during the negotiation, e.g. when new/old agents can enter/exit the environment at any time (here, *Not Tested* indicates that authors claim to support this feature, but do not test it); *incomplete information*, whether or not negotiating parties are unaware of each other's preferences; *Adaptive*, whether or not the model can adapt well to never-before-seen negotiation settings (*Not Tested* is meant as per above); *agent-agent negotiation*, as opposed to human-agent negotiation; *use of DRL*, as opposed to RL approaches that do not rely on deep learning; and *multi-issue negotiation*, as opposed to

Table 1 Comparison between RL-based negotiation strategies

Ref.	Continuous action space	Concurrent negotiations	Dynamic environment	Incomplete information	Adaptive	Agent–agent negotiation	Use of DRL	Multi-issue negotiation
[10]	–	✗	Not tested	Not specified	✓	✓	✗	✓
[49]	✗	✗	✗	✗	✓	✓	✗	✗
[54]	✗	✗	✗	✗	✓	✓	✗	✗
[24]	–	✗	Not specified	✓	Not tested	✓	✗	✓
[50]	–	✗	Not specified	✓	Not tested	✓	✗	✗
[61]	✗	✗	✓	✓	Not tested	✓	✗	✗
[12]	–	✗	Not specified	✓	✓	✓	✗	✓
[42]	✗	✗	✓	✓	Not tested	✗	✗	✓
[28]	✗	✗	Not specified	✓	Not tested	✗	✗	✓
[7]	✗	✗	✗	✓	✓	✓	✗	✓
[22]	–	✗	✓	✓	✓	✓	✗	✓
[43]	✗	✗	✗	✓	✓	✓	✓	✓
[11]	–	✗	✓	✓	✓	✓	✓	✓
[6]	✓	✗	✗	✓	✓	✓	✓	✓
[5]	✓	✓	✓	✓	✓	✓	✓	Not tested

negotiating over a single issue. Regarding the last attribute, we have tested our ANEGMA model only for single-issue negotiations, but our approach can be extended to multi-issue negotiations, as we did in [6].

3 Proposed work

In this section, we formulate the negotiation environment and introduce our agent negotiation model called *ANEGMA* (*A*daptive *NEG*otiation model for e-*MA*rkets).

3.1 Negotiation environment

We consider e-marketplaces like E-bay where the competition is visible, i.e. a buyer agent can observe the number of competitors that are dealing with the same resource from the same seller. We assume that the environment consists of a single e-market m with P agents, with a non-empty set of buyer agents B_m and a non-empty set of seller agents S_m—these sets need not be mutually exclusive. For a buyer $b \in B_m$ and resource r, we denote with $S^t_{b,r} \subseteq S_m$ the set of seller agents from market m which, at time point t, negotiate with b for a resource r over a range of issues I. The buyer agent b uses $|S^t_{b,r}|$ negotiation threads, in order to negotiate concurrently with each seller in $S^t_{b,r}$. We assume that no agent can be both buyer and seller for the same resource at the same time, that is, $\forall b, r, t.\ s \in S^t_{b,r} \implies S^t_{s,r} = \emptyset$. The set $C^t_{b,r} = \{b' \neq b \in B_m \mid S^t_{b',r} \cap S^t_{b,r} \neq \emptyset\}$ includes the competitors of b, i.e., those agents negotiating with the same sellers and for the same resource r as those of b.

We adopt the negotiation protocol of [2], since it supports concurrent bilateral negotiations, and we are interested in realistic settings where one buyer negotiates with several sellers concurrently before making a final decision. In general, a negotiation protocol describes the set of rules that each buyer b and seller s should follow during a negotiation thread, including the valid moves agents can take at any state of the negotiation. The protocol is known to all agents in advance. The protocol, illustrated in Fig. 1, assumes an open e-market environment, i.e., where agents can enter or leave the negotiation at their own will. We assume each negotiation focuses on a single resource characterized uniquely by a class and a fixed, non-negotiable, set of properties. The class ‘laptop’ with properties ‘Lenovo/16 GB RAM/500 GB HDD’ is an example of a resource r which can be used during negotiation between two agents. For such a resource r, we negotiate over a single issue, namely, price. We further assume that negotiation is represented internally for a buyer agent as a dialogue with a unique identifier so that the agent can distinguish between different negotiations for the same resource originating from different sellers. It is beyond the scope of this work to deal with multiple resources.

Furthermore, a buyer agent b always starts the negotiation by making an offer. With t_{start} we denote the start time of the negotiation, and with t_b the maximum duration of any negotiation, which for simplicity, is the same for all the agents during all the negotiation sessions irrespective of which resource these agents are negotiating for. The deadline for b is, thus, $t_{end} = t_{start} + t_b$.

Information about the deadline t_b, Initial Price IP_b and Reservation Price RP_b is private to each $b \in B_m$. Each seller s also has its own Initial Price IP_s, Reservation Price RP_s and maximum negotiation duration parameter t_s (which are not visible by other agents). The

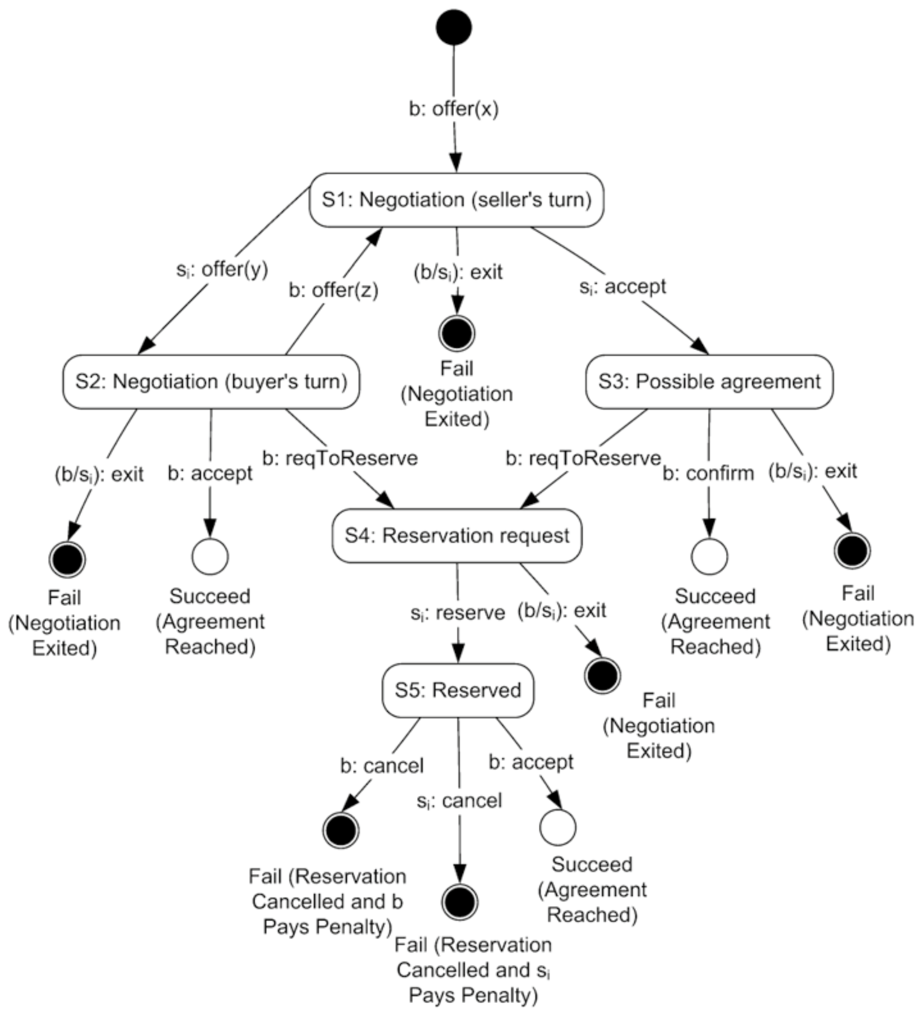
b: offer(x)
S1: Negotiation (seller's turn)
s$_i$: offer(y)
b: offer(z)
(b/s$_i$): exit
s$_i$: accept
Fail (Negotiation Exited)
S2: Negotiation (buyer's turn)
S3: Possible agreement
(b/s$_i$): exit
b: accept
b: reqToReserve
b: reqToReserve
b: confirm
(b/s$_i$): exit
Fail (Negotiation Exited)
Succeed (Agreement Reached)
S4: Reservation request
Succeed (Agreement Reached)
Fail (Negotiation Exited)
s$_i$: reserve
(b/s$_i$): exit
S5: Reserved
Fail (Negotiation Exited)
b: cancel
b: accept
s$_i$: cancel
Fail (Reservation Cancelled and b Pays Penalty)
Succeed (Agreement Reached)
Fail (Reservation Cancelled and s$_i$ Pays Penalty)

Fig. 1 Negotiation protocol [2]

protocol is turn-based and allows agents to take actions from a pool *Actions* at each negotiation state (from S1 to S5, see Fig. 1):

$$Actions = \{offer(x), reqToReserve, reserve, cancel, confirm, accept, exit\}, \tag{1}$$

where

– *offer(x)* The offer made by *b* or *s*, where *x* is the price.
– *accept* On performing this action, *b* or *s* agrees to the last offer made by their counterpart. When performed by the buyer, an *accept* leads to successful completion of the negotiation (see states S2 and S5). When performed by the seller, the buyer either acknowledges it with a *confirm* action or can buy more time with a *reqToReserve* action.

- *reqToReserve* After s makes a counter-offer (state S2) or accepts b's offer (state S3), b can perform this action to request s to reserve the resource with the latest offer. By not committing immediately to accepting the offer, b can wait for a better offer from another seller (and negotiation thread) $s' \in S^t_{b,r} \setminus \{s\}$.
- *reserve* It is used by s to acknowledge and agree to a *reqToReserve* action from b.
- *cancel* It allows both b and s to cancel their reserved offers. The cancelling agent pays a penalty to the other negotiating agent to avoid unnecessary cancellations and bias. Cancelling leads to no agreement.
- *confirm* With this action, buyer b acknowledges that the seller has accepted b's offer, and the negotiation terminates with an agreement. When dealing concurrently with different sellers for the same resource, b is allowed to send a *confirm* action only to one seller to reach an agreement.
- *exit* It allows both b and s to withdraw from the negotiation at any time (without notifying the opponent) implying that negotiation has failed.

An outcome is either *Fail* if b or s performs an *exit* or *cancel*; or it is *Succeed* if b accepts or confirms the current offer.

3.2 *ANEGMA* components

Our proposed agent negotiation model supports learning during concurrent bilateral negotiations with unknown opponents in dynamic and complex e-marketplaces. In this model, we use a centralized approach in which the coordination is done internally to the agent via multi-threading synchronization. This approach minimizes the agent communication overhead and thus, improves the run-time performance. The different components of the proposed model are shown in Fig. 2 and explained below.

3.2.1 Physical capabilities

The *sensors* of the agent enable it to access an e-marketplace. They allow a buyer b to perceive the current (external) state of the environment s_t and represent that state locally in the form of internal attributes as shown in Table 2. Some of these attributes (NS_r, NC_r) are perceived by the agent using its sensors, some of them (IP_b, RP_b, t_{end}) are stored locally in its knowledge base and some of the them (S_{neg}, X_{best}, T_{left}) are obtained while interacting with other seller agents during a negotiation. At time t, the internal agent representation of the environment is s_t, which is used by the agent to decide what action a_t to execute using its *actuators*. Action execution then changes the state of the environment to s_{t+1}.

3.2.2 Learning capabilities

The foundation of our model is a component providing learning capabilities similar to those in the Actor-Critic architecture of [31]. It consists of three sub-components: *Negotiation Experience*, *Decide* and *Evaluate*.

Negotiation Experience stores historical information about previous *negotiation experiences N* which involve the interactions of an agent with other agents in the market. Experience elements are of the form $\langle s_t, a_t, r_t, s_{t+1} \rangle$, where s_t is the internal representation of the e-market environment state perceived by the agent at time t, a_t is an action performed by

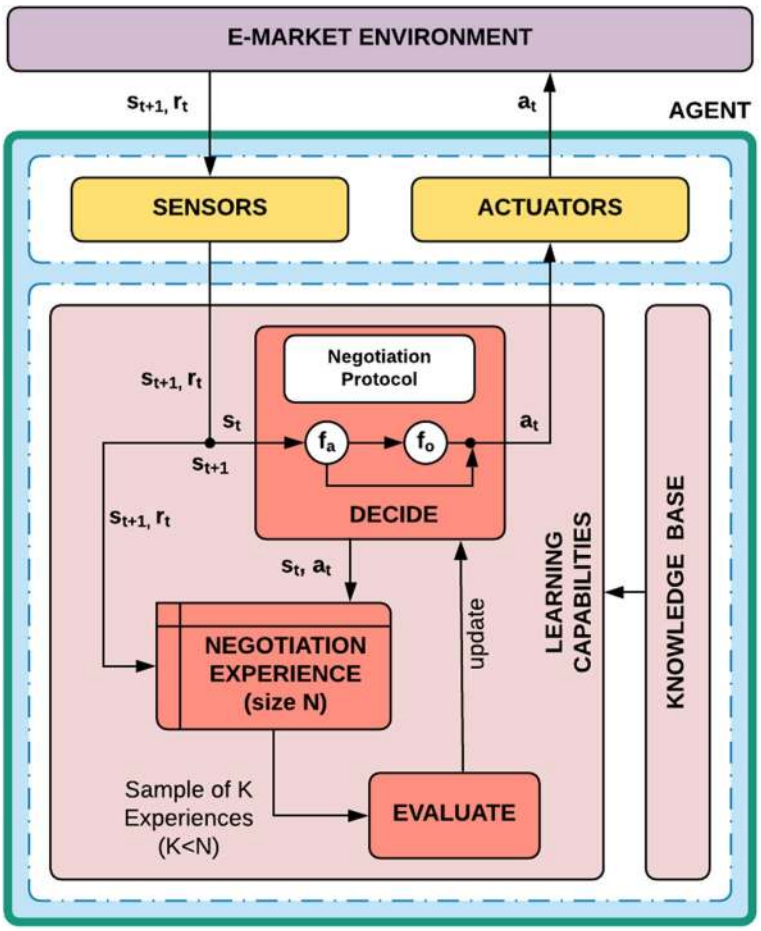

Fig. 2 The architecture of *ANEGMA*

Table 2 Agent's state attributes

Attribute	Description
NS_r	Number of sellers that b is concurrently dealing for resource r at time t ($\lvert S^t_{b,r}\rvert$)
NC_r	Number of buyer agents competing with b for resource r at time t ($\lvert C^t_{b,r}\rvert$)
S_{neg}	Current state of the negotiation protocol (S1–S5, see Fig. 1)
X_{best}	Best offer made by either b or s in S_{neg}
T_{left}	Time left for b to reach t_{end} after the last action of s
IP_b	Minimum price which b can offer at the start of the negotiation
RP_b	Maximum price which b can offer to s

b at s_t, r_t is a scalar reward or feedback received from the environment and s_{t+1} is the new e-market state after executing a_t.

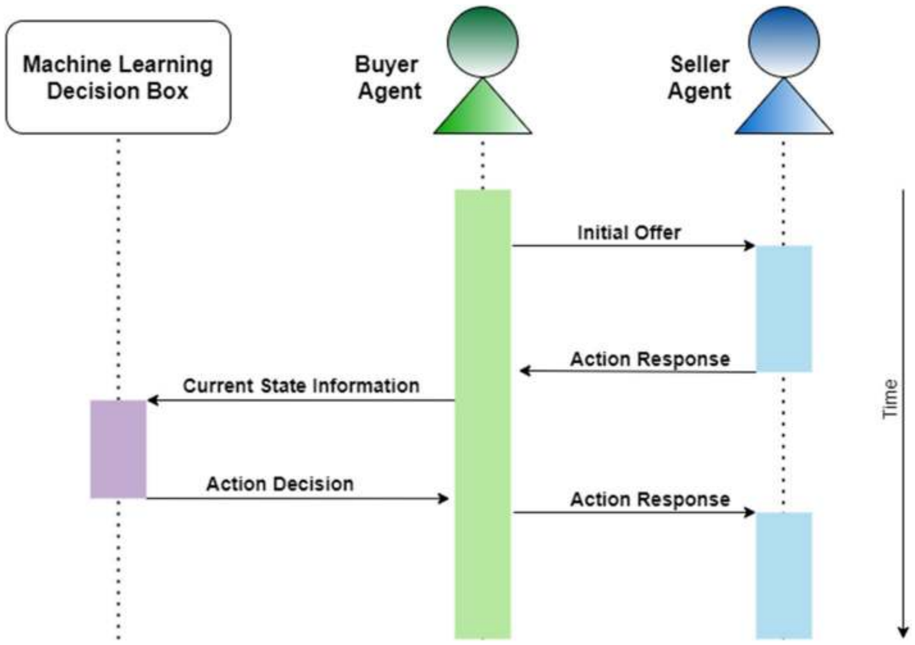

Fig. 3 Sequence diagram of *ANEGMA*

The negotiation strategy is enacted by the *decide* component. At any given state s_t, the strategy determines the optimal action for b, choosing among the available set of actions *Actions*, see (1). In particular, the strategy builds on two functions f_a and f_o. Function f_a takes state s_t as an input and returns a discrete action among *offer(x), accept, confirm, reqToReserve* and *exit*, see (2). When f_a decides to perform an *offer(x)* action, f_o is used to compute, given an input state s_t, the value of x, see (3). The functions f_a and f_o belong to Machine Learning decision box of sequence diagram presented in Fig. 3.

$$f_a(s_t) = a_t, \quad \text{where } a_t \in Actions \tag{2}$$

$$f_o(s_t) = x, \quad \text{where } x \in [IP_b, RP_b] \tag{3}$$

Evaluate refers to a critic which helps b learn and evolve the strategy for unknown and dynamic environments. It is a function of K (where $K < N$) randomly selected past negotiation experiences. The learning process of b is *retrospective* since it depends on the feedback (or scalar rewards) obtained from the e-market environment by performing action (either discrete or continuous) a_t at state s_t. Our design of reward functions accelerate agent learning by allowing b to receive rewards after every action it performs in the environment instead of receiving only at the end of the negotiation. The reward at time t, r_t is given by:

$$r_t = \begin{cases} U_b(x,t), & \text{if Succeed} \\ -1, & \text{if Fail} \\ r'_t & \text{if } a_t = \text{offer(x)} \\ 0, & \text{otherwise} \end{cases}, \text{ where} \tag{4}$$

$$r'_t = \begin{cases} U_b(x,t), & \text{if } x \leq \min(O_t) \\ -1, & \text{if } x > \min(O_t) \end{cases} \text{ and} \tag{5}$$

$$U_b(x,t) = \left(\frac{RP_b - x}{RP_b - IP_b} \right) \cdot \left(\frac{t}{t_{end}} \right)^{d_t}. \tag{6}$$

The reward values r_t and r'_t computed in (4) and (5) evaluate the discrete action decided by f_a and continuous action decided by f_o at time t respectively. Function $U_b(x,t)$, see (6), refers to the utility of offer $x = f_o(s_t)$ at time t and is calculated using Initial Price (IP_b), Reservation Price (RP_b), offer x, and a temporal discount factor $d_t \in [0,1]$ to penalize delays in negotiation, which was set to 0.6 in our experiments. Higher d_t value implies higher penalty due to delay. The reward r'_t in (5) helps b learn that it should not offer more than what active sellers have already offered: O_t is a list of preferred offers received from sellers $s \in S^t_{b,r}$ at time t, which b maintains during the negotiation. To sum up, our reward function is designed to encourage (i.e returns a positive reward value) our agent to conclude successful negotiation timely and discourage (i.e. returns a negative reward value) no deal or when our buyer agent offers more than any of the offers proposed by active sellers. Otherwise, it is neutral to all other actions (i.e. returns 0 reward).

4 Methods

In our approach, we first use SL to pre-train the *ANEGMA* agent using supervision examples collected from existing negotiation strategies. Such pre-trained strategy is then evolved via RL using experience and rewards collected while interacting with other agents in the negotiation environment. This combination of SL and RL approaches enhances the process of learning an optimal strategy. This is because applying RL alone from scratch would require a large amount of experience before reaching a reasonable strategy, which might hinder the online performance of our agent. On the other hand, starting from a pre-trained policy ensures quicker convergence (as demonstrated empirically in Sect. 5). In this section, we describe the methods for collection of supervision examples and the relevant learning techniques.

4.1 Data set collection

In order to collect the data set for pre-training the *ANEGMA* agent via SL, we have used the *RECON* simulation environment [1]. A key advantage of this solution is that we can generate arbitrarily large sets of synthetic negotiation data, and for different choices of buyer and seller strategies. While, in principle, real-world market data could be used for this purpose as well, to the best of our knowledge, no publicly available datasets exist that fit our settings. In particular, in our experiments we generate supervision data using the buyer strategies of [2] and [55] (see Sect. 5). RECON supports concurrent negotiations between buyers and seller agents and is built on the top of GOLEMlite [37],which is a Java library for managing e-markets and extract relevant negotiation statistics.

RECON consists of the following three phases as shown in Fig. 4: Phase 1 (*Configuration*) allows the user to define simulation parameters such as market density, market ratio, Zone of Agreement (ZoA), deadline, number of simulation runs, types of negotiating

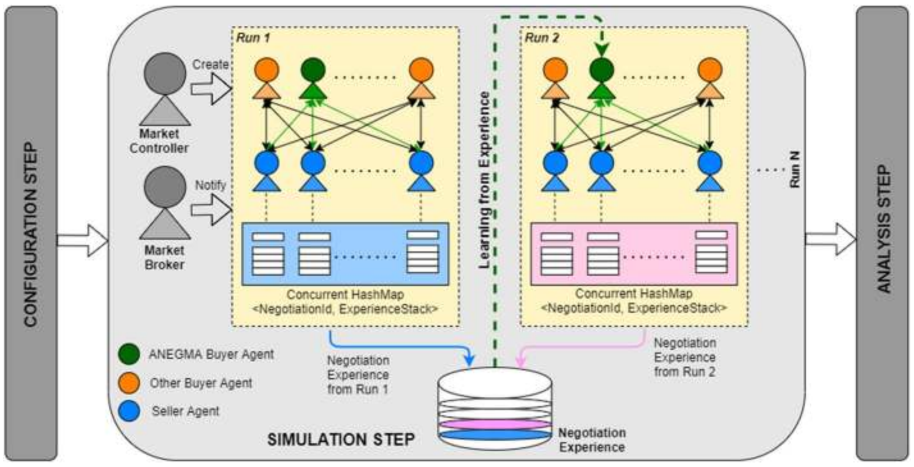

Fig. 4 Modified architecture of *RECON* simulation environment. Here, buyer agent stores the negotiation experience w.r.t. different sellers concurrently from each run in a global database (*Negotiation Experience*) to use the updated negotiation strategy (learned from past experiences) in new simulation runs

agents and their initial and reservation prices. Phase 2 (*Simulation*) conducts the actual negotiations between market agents (buyer agents and seller agents) based on the information from Phase 1 (*Configuration*). These negotiations are managed with the help of two infrastructure agents called Market Controller and Market Broker. The prime role of the Market Controller is to oversee the simulations. This involves initializing the simulations, creating the market agents and saving the negotiation logs at fixed time intervals. The market broker helps in notifying each agent about the entry of new agent in an e-market. Finally, Phase 3 (*Analysis*) analyses the negotiation logs to evaluate the performance of negotiations in terms of the various metrics defined by the user such as average utility rate and average negotiation time. We have substantially extended the RECON environment by adding a learning component to Phase 2 (*Simulation*), motivated by our *ANEGMA* model. During each simulation run, our buyer agent maintains a Concurrent Hash Map of negotiation IDs and a stack of past experiences while negotiating with different sellers concurrently, which is eventually added to a global memory *Negotiation Experience* at the end of each run. These past experiences are used by the buyer agent to learn and update the negotiation strategy and use it during new simulation runs. We have further extended RECON's market controller to handle the whole learning process during all simulations.

4.2 Strategy representation

We represent both strategies f_a and f_o (see Eqs. 2 and 3) using ANNs [18], as these are powerful function approximators and benefit from extremely effective learning algorithms. From a machine learning perspective, approximating f_a amounts to solving a *classification* problem because of f_a's discrete output domain, see (2). On the other hand, approximating f_o corresponds to solving a *regression* problem because of f_o's continuous output, see (2).

In particular, we use feed-forward neural networks, i.e., functions organized into several layers, where each layer comprises a number of neurons that process information from the previous layer. Formally, let l be the total number of layers in the network, which includes $l-1$ hidden layers and one output layer. Let n_i be the number of neurons in layer

i ($i = 1, \dots, l$), where n_0 be the number of neurons in the input layer (i.e., the input dimensionality). For input $x \in \mathbb{R}^{n_0}$, the function computed by a feed-forward neural network F is

$$F(x) = f^{(l)}\left(f^{(l-1)}\left(\dots\left(f^{(1)}(x)\right)\dots\right)\right) \tag{7}$$

where $f^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ is the function computed by the ith layer, which is given by

$$f^{(i)}(p_{i-1}) = g^{(i)}(W^{(i,i-1)} \cdot p^{(i-1)} + b^{(i)}), \quad i = 1, \dots, l \tag{8}$$

where $p_{i-1} \in \mathbb{R}^{n_{i-1}}$ is the output vector of layer $i-1$, $W^{(i,i-1)} \in \mathbb{R}^{n_i \times n_{i-1}}$ is the weight matrix connecting p_{i-1} to the neurons of layer i, $b^{(i)} \in \mathbb{R}^{n_i}$ is the bias vector of layer i, and $g^{(i)}$ is the activation function of the neurons of layer i. For our experiments, we used softmax activation function for classification and linear activation function for regression at the final output layer of our models f_a [i.e. classification (4)] and f_o [i.e. regression (5)] respectively.

Learning an ANN corresponds to finding values of its weights and biases that minimize a given loss function. The learnable parameters are typically updated via some form of gradient descent, where the gradient of the loss function w.r.t. the parameters is computed via back-propagation [18].

In supervised learning, the loss function captures the deviation between the supervision data and the corresponding model's predictions. We used cross-entropy and mean square error to approximate f_a and f_o respectively.

For each data sample $x \in \mathbb{R}^{n_0}$, the network's prediction is compared to the actual known target value of that data sample (discrete or continuous value). The function parameters (weights and biases) are also learned and modified during training so to minimize the loss. These modifications are performed in the backward direction from the output layer through each hidden layer down to the first hidden layer.

To reduce over-fitting and generalization error, during the training of the ANN we applied regularization techniques, drop-out in particular.

4.3 Reinforcement learning

During our experiments, sellers and competitor buyers use fixed strategies that are initially unknown to the buyer. As these strategies are fixed, they will be learned by *ANEGMA* later, after a number of simulation runs. Thus after a number of negotiation simulation runs our environment can be considered *fully-observable*. In addition, as our environment is dynamic (agents can leave and enter the market at any time) and episodic (the negotiation terminates at some point), we use a *model-free*, *off-policy* RL approach which generates a *deterministic policy* based on the *policy gradient* method [51] to support continuous control. More specifically, we use the *Deep Deterministic Policy Gradient algorithm (DDPG)* [30] which is an actor-critic RL approach and generates a deterministic action selection policy. We consider a model-free RL approach because our problem is how to make an agent decide what action to take next in a negotiation dialogue rather than predicting the new state of the environment. In other words, we are not learning a model of the environment, as the strategies of the sellers and the competitor buyer agents are not observable properties of the environment's state. For this, we use a market broker that notifies the negotiating parties about the number of active sellers and active competitors at a particular time. Thus, our buyer agent's emphasis is more on learning what action to take next and not the state transition function of the environment. We consider the off-policy approach (i.e. an agent attempts to evaluate or improve the policy which is different from the one that was used to take an action) for independent exploration of continuous action spaces [31].

When being in a state s_t, DDPG uses a so-called *actor* network μ to select an action a_t, and a so-called *critic* network Q to predict the value Q_t at state s_t of the action selected by the actor:

$$a_t = \mu(s_t \mid \theta^\mu) \tag{9}$$

$$Q_t(s_t, a_t \mid \theta^Q) = Q(s_t, \mu(s_t \mid \theta^\mu) \mid \theta^Q) \tag{10}$$

In (9) and (10), θ^μ and θ^Q are, respectively, the learnable parameters of the actor and critic neural networks. The parameters of the actor network are updated by the Deterministic Policy Gradient method [46]. The objective of the actor policy function is to maximize the expected return J calculated by the critic function:

$$J = \mathbb{E}[Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)}]. \tag{11}$$

To this purpose, the parameters of μ are updated (via gradient ascent) using the gradient of J w.r.t. the actor policy parameters. In particular, the expectation in (11) is approximated using the average of K randomly selected past experiences (or mini-batches) (s_i, a_i, r_i, s_{i+1}).

$$\nabla_{\theta^\mu} J \approx \frac{1}{K} \sum_{i=1}^{K} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i} \right] \tag{12}$$

The critic network Q should predict the expected return obtained by performing action a_t at state s_t and thereafter follow the policy entailed by the actor μ. For this purpose, [30] shows that Q can be derived, using K random mini-batches, by minimizing the following loss function:

$$L = \frac{1}{K} \sum_{i=1}^{K} (y_i - Q(s_i, a_i|\theta^Q))^2, \text{ where} \tag{13}$$

$$y_i = r_i + \gamma Q(s_{i+1}, \mu(s_{i+1}|\theta^\mu)|\theta^Q), \tag{14}$$

and $\gamma \in (0, 1)$ is a discount factor. Since the target Q-value y_i used to update Q depends on Q itself, which might cause divergence, DDPG employs two additional neural networks called actor target network μ' and critic target network Q' in place of μ and Q in (14). These are copies of μ and Q which are updated in a soft manner, i.e., by slowing tracking μ and Q rather than exactly copying them, which the effect of regularizing learning and increasing stability. See [30] for further details.

5 Experimental setup and results

In this section, we experimentally evaluate our *ANEGMA* approach in negotiations against unknown opponents during concurrent bilateral negotiations in different e-market settings.

Table 3 Performance evaluation metrics

Metric	Definition	Ideal value
U_{avg}	Total negotiation utility averaged over the successful negotiations	High (1.0)
T_{avg}	Total time taken by the buyer agent (in milliseconds) averaged over all successful negotiations to reach the agreement	Low ($\approx$ 1000 ms)
$S_{\%}$	Proportion of successful negotiations	High (100%)

5.1 Experimental settings

We consider the following buyer strategies:

- *CONAN* [2] A heuristic strategy which uses a weighted combination of agent's internal state attributes as well as environmental parameters (in which agent situates) to calculate the concession rate. This strategy lets the agent negotiate with multiple sellers concurrently without the help of an additional coordinator.
- *Williams* [55] A strategy performing Gaussian process for predicting the seller agent's future utility while bidding a counter-offer. This strategy is originally used to negotiate with multiple opponents for the same item over multiple issues with the help of a coordinator. The coordinator is responsible for finding the best of all deals with different opponents based on time and utility.
- *SL-C*[2] An ANN-based strategy obtained using supervised learning from CONAN data.
- *SL-W*[2] An ANN-based strategy obtained using supervised learning from Williams' data.
- *DRL* A DRL strategy initialized with an ANN with random parameters.
- *ANEGMA-C* Our ANEGMA strategy obtained via DRL and initialized with the ANN pre-trained with CONAN data (SL-C).
- *ANEGMA-W* Our ANEGMA strategy obtained via DRL and initialized with the ANN pre-trained with Williams data (SL-W).

For carrying out the experiments, we have used the RECON simulation environment [1] and extended it to support online agent learning as shown in Fig. 4.

Performance evaluation measures To successfully evaluate the performance of *ANEGMA* (ANEGMA-C and ANEGMA-W) and compare it with other negotiation approaches, we selected the following widely adopted metrics [2, 15, 39, 55]: *Average utility rate (U_{avg})*, *Average negotiation time (T_{avg})* and *Percentage of successful negotiations ($S_{\%}$)*, which are described in Table 3.

Seller strategies We consider two widely-known and standard groups of fixed seller strategies developed by Faratin [15]: Time-Dependent and Behaviour-Dependent, each consisting of three different types of seller strategies. In time-dependent strategies (*Linear*, *Conceder* and *Boulware*), the seller considers the remaining negotiation time for calculating the counter-offer value and the acceptance value for the offer received from the buyer. On the other hand, in behaviour-dependent strategies (*Relative tit-for-tat*, *Random Absolute tit-for-tat* and *Averaged tit-for-tat*), the seller imitates

[2] SL-X is identical to pre-training phase of ANEGMA-X.

Table 4 Simulation parameter values

Parameter	Values range		
	100% *ZoA* (high)	60% *ZoA* (average)	10% *ZoA* (low)
IP_b	[300–350]	[300–350]	[300–350]
RP_b	[500–550]	[500–550]	[500–550]
IP_s	[500–550]	[580–630]	[680–730]
RP_s	[300–350]	[380–430]	[480–530]
MD	$\{30, 40, 50\}$	$\{18, 23, 28\}$	$\{8, 10, 12\}$
MR	$\{10:1,\ 1:1,\ 1:10\}$	$\{5:1,\ 1:1,\ 1:5\}$	$\{2:1,\ 1:1,\ 1:2\}$
t_{end}	[151–210 s]	[91–150 s]	[30–90 s]

the observed behaviour of the buyers in order to compute the counter-offer. During experimentation, the same private deadlines were used for both sellers and buyer. Other parameters such as IP_s and RP_s are determined by the *ZoA* parameter, as shown in Table 4.

Competitor strategies All competitor strategies are chosen randomly between Simple Buyer (which generates offers randomly) and Nice Tit for Tat (which reproduces the opponent's behaviours of the previous negotiation rounds by reciprocating the opponent's concessions).

Simulation parameters We assume that the buyer negotiates with multiple sellers concurrently to buy a second-hand laptop ($r = Laptop$) based only on a single issue *Price* ($I = \{Price\}$). We stress that the single-issue assumption is not unrealistic for e-markets like e-Bay, where sellers advertise a product with a fixed set of issues (e.g. Lenovo, 16 GB RAM, 250 GB HDD, i7 processor) and the only issue being negotiated is price. The simulated market allows the agents to enter and leave the market at their own will. The maximum number of agents allowed in the market, the demand/supply ratio, the buyer's deadline and the *ZoA*s are simulation-dependent.

As in [2], three qualitative values are considered for each parameter during simulations, e.g., High, Average and Low for *MD* or t_{end}. Parameters are reported in Table 4. The user can select one of such qualitative values for each parameter. Each qualitative value corresponds to a set of three quantitative values, of which only one is chosen at random for each simulation (e.g., setting *High* for parameter *MD* corresponds to choosing at random among 30, 40, and 50). The only exception is parameter *ZoA*, which maps to a range of uniformly distributed quantitative values for the seller's initial price IP_s and reservation price RP_s (e.g., selecting *Average* for *ZoA* leads to a value of IP_s uniformly sampled in the interval [580, 630]). Therefore, the total number of simulation settings is 81, as we consider 3 possible settings for each of *MD*, *MR*, t_{end}, and *ZoA* (see Table 4).

Neural network architecture We represent the supervised learning policy as a neural network with 2 fully-connected hidden layers of 64 units and one output layer. The hidden layers use ReLU (Rectified Linear Unit) activation function whereas the output layer uses softmax and linear activation functions for classification and regression respectively. For DDPG, we represent deep neural networks with the same above mentioned neural networks.

5.2 Experimental hypotheses

With our experiments, we aim to demonstrate the following hypotheses:

Hypothesis A The *Market Density (MD)*, the *Market ratio* or *Demand/Supply Ratio* (*MR*), the *Zone of Agreement* (*ZoA*) and the *Buyer's Deadline* (t_{end}) have a considerable effect on the success of negotiations. Here,

- *MD* is the total agents in the e-market at any given time dealing with the same resource as that of our buyer.
- *MR* is the ratio of the total number of buyers over the sellers in the e-market.
- *ZoA* refers to the intersection between the price ranges of buyers and sellers for them to agree.

In practice, buyers have no control over these parameters except the deadline (t_{end}), which can be decided by the user, or constrained by a higher-level goal the buyer is trying to achieve. While this hypothesis is not directly concerned with the performance of ANEGMA, it establishes that, for an adequate performance evaluation, it is necessary to fix a particular choice of these parameters. Otherwise, the performance variability will be too high to make any useful assessment.

Hypothesis B The *ANEGMA* buyer outperforms the SL-only, DRL-only, CONAN, and Williams' negotiation strategies in terms of U_{avg}, T_{avg} and $S_{\%}$ in a range of e-market settings.

Hypothesis C An *ANEGMA* buyer if trained against a specific seller strategy, still performs well against other unknown seller strategies. This shows that the *ANEGMA* agent behaviour is *adaptive* in the sense that the agent transfers knowledge from previous experience to unknown e-market settings.

5.3 Empirical evaluation

We evaluate and discuss the three research hypotheses introduced at the beginning of the section.

5.3.1 Hypothesis A (*MD*, *MR*, *ZoA* and t_{end} have significant impact on negotiations)

We experimented with 81 different e-market settings over 500 simulations using the CONAN buyer strategy. Both time-dependent and behaviour-dependent seller strategies were considered for each setting. These experiments suggest that *MD* and *ZoA* have a considerable effect on $S_{\%}$ (Figs. 5, 6, 7, 8). In Fig. 5, we observe that the agents reach more negotiation agreements when *MD* is low. This suggests that in small markets offering the required resource, the number of successful deals is maximised, which in turn implies that being in a large market isn't always better. Also, there is not much difference in the agreement rate for 60% and 100% *ZoA* when *MD* is low. The small number of successful negotiations for 10% *ZoA* is not unexpected since only a minority of agents is willing to concede more in such a small *ZoA*. On the other hand, *MR* and t_{end} have , according to our experiments, a comparably minor impact on the negotiation success (see Figs. 7, 8). Also, only some effect of *MR* on $S_{\%}$ is observed under low *MD* against behaviour-dependent strategies

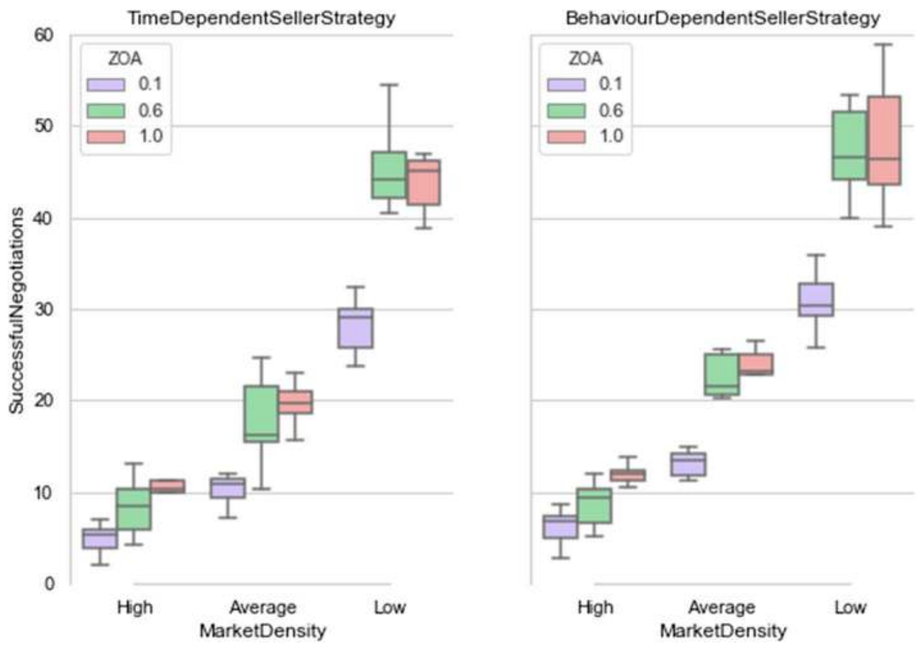

Fig. 5 Effect of market density and ZoA on percentage of successful negotiations

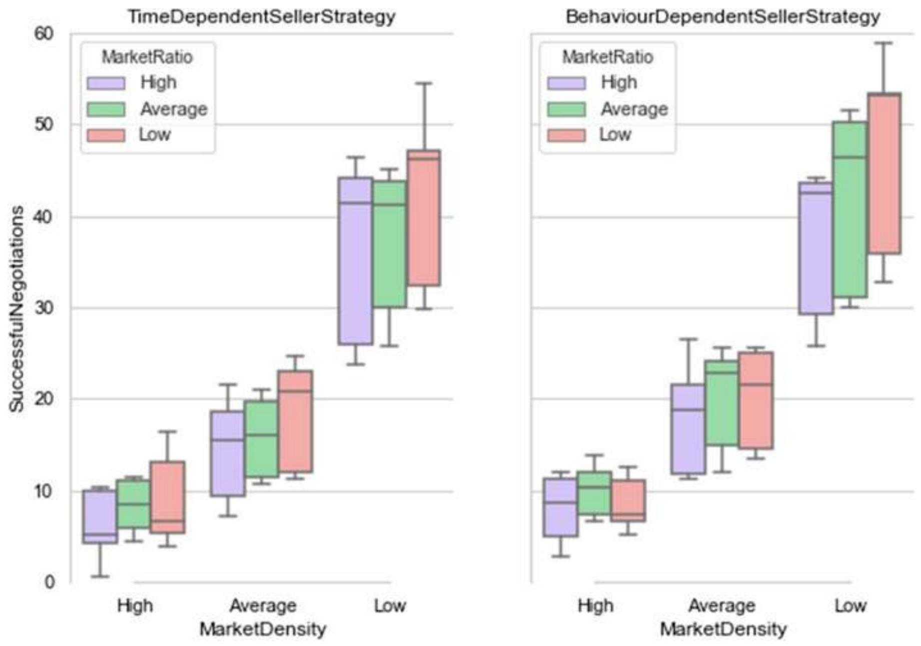

Fig. 6 Effect of market density and market ratio on percentage of successful negotiations

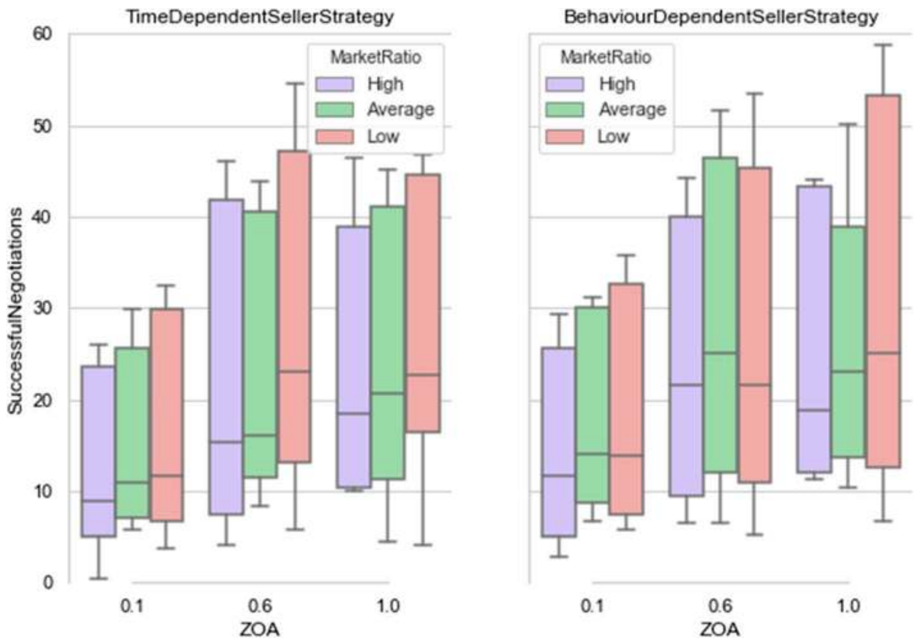

Fig. 7 Effect of market ratio and ZoA on percentage of successful negotiations

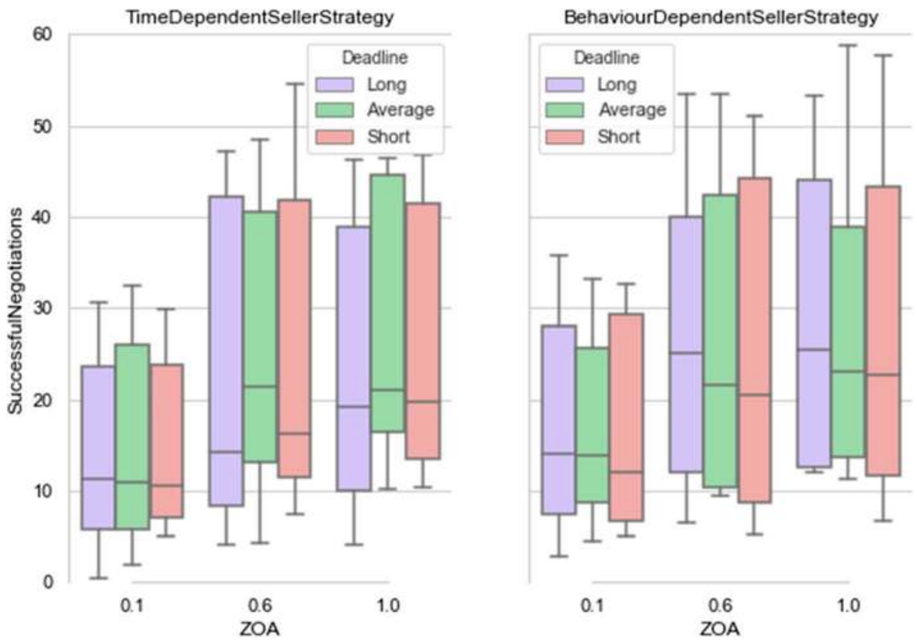

Fig. 8 Effect of deadline and ZoA on percentage of successful negotiations

Table 5 Performance comparison of CONAN and Williams' model for both 60% and 100% *ZOA*

Metric	CONAN		Williams'	
Conceder time dependent seller strategy				
	60% ZoA	*100% ZoA*	*60% ZoA*	*100% ZoA*
U_{avg}	$\mathbf{0.27 \pm 0.03}$	$\mathbf{0.25 \pm 0.07}$	0.18 ± 0.08	0.17 ± 0.04
T_{avg}	$\mathbf{172942.78 \pm 15177.77}$	174611.43 ± 15139.52	177091.09 ± 15304.90	$\mathbf{174468.31 \pm 15365.11}$
$S_{\%}$	**80.80**	**79.00**	78.20	78.00
Relative tit for tat behaviour seller strategy				
U_{avg}	$\mathbf{0.25 \pm 0.03}$	$\mathbf{0.24 \pm 0.04}$	0.22 ± 0.05	0.21 ± 0.06
T_{avg}	$\mathbf{176018.69 \pm 14380.28}$	179529.47 ± 14165.15	176334.65 ± 14683.03	$\mathbf{176468.31 \pm 15365.11}$
$S_{\%}$	**81.80**	**79.80**	73.00	73.21

Best results are in bold

Table 6 Training accuracy's of ANN (in %) when trained using datasets collected by negotiating CONAN (i.e. SL-C) and Williams' (i.e. SL-W) buyer strategy (for different *ZoA*s) against time-dependent and behaviour-dependent seller strategies

ZOA (%)	Conceder time dependent		Relative tit for tat behaviour dependent	
	SL-C	SL-W	SL-C	SL-W
10	93.88	94.73	94.65	94.77
60	97.65	97.88	97.68	97.86
100	95.96	96.23	96.88	95.73

as shown in Fig. 6). Moreover, we performed significance tests (i.e. Z-tests for independent proportions) for all the relevant pair-wise comparisons. All the differences in the proportions of successful runs were found significant at $p < 2.12E - 13$.[3]

5.3.2 Hypothesis B (ANEGMA outperforms SL, CONAN and Williams')

We performed simulations for our *ANEGMA* agent in low *MD*, 60% and 100% *ZoA*, high *MR*, and a long t_{end} because these settings yielded the best performance in terms of $S_{\%}$ in our experiments for Hypothesis A. To test how our strategy learns against two different categories of fixed seller strategies (i.e. Time-dependent and Behaviour-dependent) as well as to limit the experiments, we randomly choose *Conceder Time Dependent* and *Relative Tit for Tat Behaviour Dependent* seller strategies in the above simulation settings.

Firstly, we collected training data for ANN using two distinct strategies for supervision, viz. CONAN [2] and Williams' [55]. Both were run for 500 simulations and with the same settings. Table 5 compares the performances of CONAN's and Williams' models. CONAN outperforms Williams' strategy in these settings in terms of U_{avg} and $S_{\%}$.

Then, the resulting trained ANN models—called SL-C and SL-W respectively—were used as the initial strategies in our DRL approach (based on DDPG), where strategies

[3] For each ZoA=H,A,L, we tested (MD=H vs. MD=A) and (MD=A vs. MD=L). For each MD=H,A,L, we tested (ZoA=L vs. ZoA=A) and (ZoA=L vs. ZoA=H).

Table 7 Performance comparison of SL VS ANEGMA VS DRL when *ZoA* is 60%

Metric	*SL-C*	*SL-W*	*ANEGMA-C*	*ANEGMA-W*	DRL
Trained and tested on conceder time dependent seller strategy					
U_{avg}	0.27 ± 0.04	0.21 ± 0.08	$\mathbf{0.29 \pm 0.04}$	0.21 ± 0.04	-0.38 ± 0.14
T_{avg}	$173,529.47 \pm 14,651.15$	$171,096.09 \pm 14,584.90$	$67,750.62 \pm 37,628.57$	$132,477.71 \pm 26,601.48$	$\mathbf{768.55 \pm 373.65}$
$S_{\%}$	81.18	80.19	**87.12**	81.19	64.36
Trained and tested on relative tit for tat behaviour dependent seller strategy					
U_{avg}	0.26 ± 0.03	0.23 ± 0.05	$\mathbf{0.29 \pm 0.03}$	0.23 ± 0.14	-0.19 ± 0.42
T_{avg}	$167,183.62 \pm 13,388.30$	$169,334.65 \pm 12,389.03$	$36,331.34 \pm 70,247.33$	$41,225.17 \pm 72,938.79$	$\mathbf{755.74 \pm 292.29}$
$S_{\%}$	82.18	75.24	**85.15**	74.26	61.38

Best results are in bold. SL-C and SL-W correspond to ANN trained using data set collected from CONAN and Williams' approach respectively, whereas ANEGMA-C and ANEGMA-W correspond to DRL initialized with SL-C and SL-W respectively

Table 8 Performance comparison of SL VS ANEGMA VS DRL when *ZoA* is 100%

Metric	*SL-C*	*SL-W*	*ANEGMA-C*	*ANEGMA-W*	DRL
Trained and tested on conceder time dependent seller strategy					
U_{avg}	0.23 ± 0.04	0.17 ± 0.08	$\mathbf{0.27 \pm 0.51}$	0.21 ± 0.71	-0.88 ± 0.16
T_{avg}	$172,234.73 \pm 14,516.15$	$170,969.09 \pm 14,464.09$	$171,266.64 \pm 11,573.38$	$185,425.74 \pm 19,909.06$	$\mathbf{1021.95 \pm 771.47}$
$S_{\%}$	77.23	76.24	**78.22**	72.28	57.42
Trained and tested on relative tit for tat behaviour dependent seller strategy					
U_{avg}	0.26 ± 0.30	0.18 ± 0.55	$\mathbf{0.29 \pm 0.35}$	0.23 ± 0.84	-0.24 ± 0.55
T_{avg}	$160,178.98 \pm 14,809.18$	$163,943.05 \pm 12,895.03$	$33,695.16 \pm 64,292.37$	$23,528.25 \pm 61,440.37$	$\mathbf{817.67 \pm 523.67}$
$S_{\%}$	73.27	72.28	**79.21**	71.81	56.43

Best results are in bold. SL-C and SL-W correspond to ANN trained using data set collected from CONAN and Williams' approach respectively, whereas ANEGMA-C and ANEGMA-W correspond to DRL initialized with SL-C and SL-W respectively

evolved using negotiation experience from additional 500 simulations. In the remainder, we will abbreviate these models by *ANEGMA-C* and *ANEGMA-W* respectively.

Finally, we used test data from 101 simulations involving online learning to compare the performance of such derived *ANEGMA-C* and *ANEGMA-W* buyers against CONAN, Williams', SL-C, SL-W, and the so-called DRL model which used DDPG but initialized with a random strategy.

According to our results shown in Tables 7 and 8, the performance of SL-C is comparable to that of CONAN for both 60% and 100% *ZoA*s (see Table 5). We observe the same for SL-W and the William's strategy. So, we conclude that our approach can successfully produce ANN strategies which are able to imitate the behaviour and performance of the CONAN and Williams' models (the training accuracies were in the range between 93.0% and 98.0% as shown in Table 6).

Even more importantly, the results demonstrate that *ANEGMA-C* (i.e. DDPG initialized with SL-C) and *ANEGMA-W* (i.e. DDPG initialized with SL-W) improve on their respective initial ANN strategies obtained by SL, and outperform DRL initialized at random for both 60% and 100% *ZoA*s, see Tables 7 and 8. This proves that both the evolution of the strategies via DRL and the initial supervision are beneficial. Furthermore, *ANEGMA-C* and *ANEGMA-W* also outperform the existing "teacher strategies" (CONAN and Williams') in terms of U_{avg} used for the initial supervision and hence can improve on them, see Table 5.

Moving further, we observe that our agent ANEGMA becomes selective and learns to focus on how to obtain maximum utility from the end agreement (by accepting or proposing a bid only if a certain dynamic threshold utility is met). Thus, the successful negotiation rate is lower as compared to SL agents that seek to maximize the average utility rate. This could be a reason why SL-W seems to outperform ANEGMA-W in terms of successful negotiation rate. Although we could incorporate the number of successful negotiations in the reward function to bias our learning to optimize this metric, we have opted for the simple and commonly used reward function related to utility value only.

5.3.3 Hypothesis C *(ANEGMA is Adaptable)*

In this final test, we evaluate how well our *ANEGMA* agents can adapt to environments different from those used at training-time. Specifically, we deploy strategies trained using *Conceder Time Dependent* opponents into an environment with *Relative Tit for Tat Behaviour Dependent* opponents, and vice-versa. The *ANEGMA* agents use experience from 500 simulations to adapt to the new environment. Results are presented in Table 9 and show clear superiority of the *ANEGMA* agents over the SL-C and SL-W strategies which, without online retraining, cannot maintain their performance in the new environment. This confirms our hypothesis that *ANEGMA* agents can learn to adapt at run-time to different unknown seller strategies.

5.3.4 Further discussion

Pondering over the negative average utility of DRL (see Tables 7, 8), recall that we define utility as in Eq. (6) but without the discount factor. Therefore, if an agent concedes a lot to make a deal, it will collect negative utility. This is precisely what happens to the initial random (and inefficient) strategy used in DRL. The combination of SL and DRL prevents this problem as it uses an initial pre-trained strategy which is much less likely to incur negative utility. For the same reason, we observe a consistently shorter T_{avg} for DRL caused by a

Table 9 Performance comparison for the adaptive behaviour of SL VS ANEGMA VS DRL when ZoA is 60%

Metric	*SL-C*	*SL-W*	*ANEGMA-C*	*ANEGMA-W*	*DRL*
Trained on relative tit for tat behaviour dependent and tested on conceder time dependent seller strategy					
U_{avg}	0.16 ± 0.05	0.17 ± 0.04	$\mathbf{0.26 \pm 0.06}$	0.23 ± 0.07	-0.36 ± 0.12
T_{avg}	$174,139.30 \pm 14,655.42$	$174,035.91 \pm 14,627.59$	$38,402.78 \pm 64,367.45$	$108,051.11 \pm 57,755.84$	$\mathbf{738.55 \pm 279.65}$
$S_{\%}$	70.29	69.30	**86.13**	81.19	54.45
Trained on conceder time dependent and tested on relative tit for tat behaviour dependent seller strategy					
U_{avg}	0.25 ± 0.05	0.21 ± 0.04	$\mathbf{0.28 \pm 0.01}$	0.21 ± 0.08	-0.28 ± 0.51
T_{avg}	$176,048.05 \pm 14,423.36$	$175,170.19 \pm 14,623.53$	$19,295.84 \pm 53,767.54$	$114,510.00 \pm 64,667.79$	$\mathbf{806.83 \pm 375.51}$
$S_{\%}$	79.21	76.23	**84.16**	71.28	51.48

Best results are in bold. SL-C and SL-W correspond to ANN trained using data set collected from CONAN and Williams' approach respectively, whereas ANEGMA-C and ANEGMA-W correspond to DRL initialized with SL-C and SL-W respectively

buyer that concedes more to reach the agreement without negotiating for a long time with the seller. Hence, a shorter T_{avg} alone does not generally imply a better negotiation performance. An additional advantage of our approach is that it alleviates the common limitation of RL, namely, that an RL agent needs a non-trivial amount of experience before reaching satisfactory performance.

5.3.5 Results summary

In this sub-section, we summarize the results from Tables 7 to 9. When ZoA is 60% and 100%, ANEGMA-C outperforms all other strategies in comparison w.r.t U_{avg} and $S_{\%}$. However, DRL outperforms in terms of T_{avg}. We have also shown the results for the adaptive behaviour of ANEGMA when ZoA is 60%, which also reflects the same outcomes, i.e. ANEGMA-C outperforms all other agents in terms of average utility rate and number of successful negotiations.

6 Conclusions and future work

ANEGMA is a novel agent negotiation model supporting agent learning and adaptation during concurrent bilateral negotiations for an important class of e-markets. An *ANEGMA* agent derives an initial neural network strategy via SL from well-known negotiation models, and evolves the strategy via DRL. We have empirically evaluated the performance of an *ANEGMA* buyer agent against fixed but unknown to the agent seller strategies in different e-market settings. We have shown that *ANEGMA* outperforms well-known "teacher strategies", the strategies trained with SL only and those trained with DRL only. Crucially, our model has also exhibited adaptive behaviour in that it can transfer to environments with unknown sellers, viz., sellers that use different strategies from those used during training.

An open problem arising from our work is how to learn transferable strategies for concurrent bilateral negotiations over multiple issues and against adaptive opponents.

References

1. Alrayes, B., Kafalı, Ö., & Stathis, K. (2016). Recon: A robust multi-agent environment for simulating concurrent negotiations. In *Recent advances in agent-based complex automated negotiation* (pp. 157–174). Springer.
2. Alrayes, B., Kafalı, Ö., & Stathis, K. (2018). Concurrent bilateral negotiation for open e-markets: The Conan strategy. *Knowledge and Information Systems, 56*(2), 463–501.

3. An, B., Sim, K. M., Tang, L. G., Li, S. Q., & Cheng, D. J. (2006). Continuous-time negotiation mechanism for software agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*(6), 1261–1272.
4. Baarslag, T., Hindriks, K., Hendrikx, M., Dirkzwager, A., & Jonker, C. (2014). Decoupling negotiating agents to explore the space of negotiation strategies. In *Novel insights in agent-based complex automated negotiation* (pp. 61–83). Springer.
5. Bagga, P., Paoletti, N., & Alrayes Bedour Stathis, K. (2020). A deep reinforcement learning approach to concurrent bilateral negotiation. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence* (pp. 297–303).
6. Bagga, P., Paoletti, N., & Stathis, K. (2020). Learnable strategies for bilateral agent negotiation over multiple issues. arXiv preprint arXiv:2009.08302.
7. Bakker, J., Hammond, A., Bloembergen, D., & Baarslag, T. (2019). Rlboa: A modular reinforcement learning framework for autonomous negotiating agents. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 260–268). International Foundation for Autonomous Agents and Multiagent Systems.
8. Bala, M. I., Vij, S., & Mukhopadhyay, D. (2013). Intelligent agent for prediction in e-negotiation: An approach. In *2013 International conference on cloud & ubiquitous computing & emerging technologies* (pp. 183–187). IEEE.
9. Buffett, S., & Spencer, B. (2007). A Bayesian classifier for learning opponents' preferences in multi-object automated negotiation. *Electronic Commerce Research and Applications, 6*(3), 274–284.
10. Cardoso, H. L., & Oliveira, E. (2000). Using and evaluating adaptive agents for electronic commerce negotiation. In *Advances in artificial intelligence* (pp. 96–105). Springer.
11. Chang, H. C. H. (2020). Multi-issue negotiation with deep reinforcement learning. *Knowledge-Based Systems,* 106544.
12. Chen, L., Dong, H., & Zhou, Y. (2014). A reinforcement learning optimized negotiation method based on mediator agent. *Expert Systems with Applications, 41*(16), 7630–7640.
13. Choi, S. P., Liu, J., & Chan, S. P. (2001). A genetic agent-based negotiation system. *Computer Networks, 37*(2), 195–204.
14. Choudhary, N., & Bharadwaj, K. (2018). Evolutionary learning approach to multi-agent negotiation for group recommender systems. *Multimedia Tools and Applications,* 1–23.
15. Faratin, P., Sierra, C., & Jennings, N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems, 24*(3–4), 159–182.
16. Fatima, S., Kraus, S., & Wooldridge, M. (2014). *Principles of automated negotiation*. Cambridge: Cambridge University Press.
17. Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2004). An agenda-based framework for multi-issue negotiation. *Artificial Intelligence, 152*(1), 1–45.
18. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
19. Hendrikx, M. (2012). Evaluating the quality of opponent models in automated bilateral negotiations.
20. Hindriks, K., & Tykhonov, D. (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* (Vol. 1, pp. 331–338).
21. Hopkins, J., Kafali, O., Alrayes, B., & Stathis, K. (2019). PIRASA: Strategic protocol selection for e-commerce agents. *Electronic Markets, 29*(2), 239–252.
22. Imran, K., Zhang, J., Pal, A., Khattak, A., Ullah, K., & Baig, S. M. (2020). Bilateral negotiations for electricity market by adaptive agent-tracking strategy. *Electric Power Systems Research, 186,* 106390.
23. Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J., & Sierra, C. (2001). Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation, 10*(2), 199–215.
24. Jian, L. (2008). An agent bilateral multi-issue alternate bidding negotiation protocol based on reinforcement learning and its application in e-commerce. In *2008 International symposium on electronic commerce and security* (pp. 217–220). IEEE.
25. Kiruthika, U., Somasundaram, T. S., & Raja, S. K. S. (2020). Lifecycle model of a negotiation agent: A survey of automated negotiation techniques. *Group Decision and Negotiation,* 1–24.
26. Kraus, S. (2001). Automated negotiation and decision making in multiagent environments. In *ECCAI advanced course on artificial intelligence* (pp. 150–172). Springer.

27. Lau, R. Y., Tang, M., Wong, O., Milliner, S. W., & Chen, Y. P. P. (2006). An evolutionary learning approach for adaptive negotiation agents. *International Journal of Intelligent Systems, 21*(1), 41–72.
28. Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., & Batra, D. (2017). Deal or no deal? end-to-end learning for negotiation dialogues. arXiv preprint arXiv:1706.05125.
29. Li, J., & Cao, Y. D. (2004). Bayesian learning in bilateral multi-issue negotiation and its application in mas-based electronic commerce. In *Proceedings. IEEE/WIC/ACM international conference on intelligent agent technology IAT 2004)* (pp. 437–440). IEEE.
30. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
31. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., & Wierstra, D.P. (2017). Continuous control with deep reinforcement learning. US Patent App. 15/217,758.
32. Lin, R., & Kraus, S. (2010). Can automated agents proficiently negotiate with humans? *Communications of the ACM, 53*(1), 78–88.
33. Lin, R., Kraus, S., Wilkenfeld, J., & Barry, J. (2006). An automated agent for bilateral negotiation with bounded rational agents with incomplete information. *Frontiers in Artificial Intelligence and Applications, 141,* 270.
34. Lomuscio, A. R., Wooldridge, M., & Jennings, N. R. (2003). A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation, 12*(1), 31–56.
35. Mansour, K., & Kowalczyk, R. (2014). Coordinating the bidding strategy in multi-issue multi-object negotiation with single and multiple providers. *IEEE Transactions on Cybernetics, 45*(10), 2261–2272.
36. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
37. Munim, A. (2013). GOLEMLite: A framework for the development of agent-based applications. Master's thesis, Royal Holloway, University of London.
38. Narayanan, V., & Jennings, N. R. (2006). Learning to negotiate optimally in non-stationary environments. In *International workshop on cooperative information agents* (pp. 288–300). Springer.
39. Nguyen, T. D., & Jennings, N. R. (2004). Coordinating multiple concurrent negotiations. In *Proceedings of the third international joint conference on autonomous agents and multiagent systems* (Vol. 3, pp. 1064–1071). IEEE Computer Society.
40. Oliver, J. R. (1996). A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of management information systems, 13*(3), 83–112.
41. Oshrat, Y., Lin, R., & Kraus, S. (2009). Facing the challenge of human-agent negotiations via effective general opponent modeling. In *Proceedings of the 8th international conference on autonomous agents and multiagent systems* (Vol. 1, pp. 377–384). International Foundation for Autonomous Agents and Multiagent Systems.
42. Papangelis, A., & Georgila, K. (2015). Reinforcement learning of multi-issue negotiation dialogue policies. In *Proceedings of the 16th annual meeting of the special interest group on discourse and dialogue* (pp. 154–158).
43. Razeghi, Y., & Yavuz, C. O. B., & Aydoğan R. . (2020). Deep reinforcement learning for acceptance strategy in bilateral negotiations. *Turkish Journal of Electrical Engineering & Computer Sciences, 28*(4), 1824–1840.
44. Ren, Z., & Anumba, C. J. (2002). Learning in multi-agent systems: A case study of construction claims negotiation. *Advanced Engineering Informatics, 16*(4), 265–275.
45. Rubinstein, A. (1982). Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society,* 97–109.
46. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31st international conference on machine learning* (pp. 387–395).
47. Sim, K. M., Guo, Y., & Shi, B. (2007). Adaptive bargaining agents that negotiate optimally and rapidly. In *2007 IEEE congress on evolutionary computation* (pp. 1007–1014). IEEE.
48. Sim, K. M., Guo, Y., & Shi, B. (2008). Blgan: Bayesian learning and genetic algorithm for supporting negotiation with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(1), 198–211.
49. Sridharan, M., & Tesauro, G. (2002). Multi-agent q-learning and regression trees for automated pricing decisions. In *Game theory and decision theory in agent-based systems* (pp. 217–234). Springer.
50. Sun, T., Zhu, Q., Xia, Y., & Cao, F. (2011). A bilateral price negotiation strategy based on Bayesian classification and q-learning. *Journal of Information & Computational Science, 8*(13), 2773–2780.

51. Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems,* 1057–1063.
52. Sycara, K., Zeng, D., et al. (1997). Benefits of learning in negotiation. In *Proceedings of the AAAI national conference on artificial intelligence* (pp. 36–41). Menlo Park, California
53. Tesauro, G. (2000). Pricing in agent economies using neural networks and multi-agent q-learning. In *Sequence learning* (pp. 288–307). Springer.
54. Tesauro, G., & Kephart, J. O. (2002). Pricing in agent economies using multi-agent q-learning. *Autonomous Agents and Multi-agent Systems, 5*(3), 289–304.
55. Williams, C. R., Robu, V., Gerding, E. H., & Jennings, N. R. (2012). Negotiating concurrently with unknown opponents in complex, real-time domains. In *Proceedings of the 20th European conference on artificial intelligence*.
56. Yu, C., Ren, F., & Zhang, M. (2013). An adaptive bilateral negotiation model based on bayesian learning. In *Complex automated negotiations: Theories, models, and software competitions* (pp. 75–93). Springer.
57. Zeng, D., & Sycara, K. (1998). Bayesian learning in negotiation. *International Journal of Human-Computer Studies, 48*(1), 125–141.
58. Zhang, J., Ren, F., & Zhang, M. (2015). Bayesian-based preference prediction in bilateral multi-issue negotiation between intelligent agents. *Knowledge-Based Systems, 84,* 108–120.
59. Zhang, M., Tan, Z., Zhao, J., & Li, L. (2008): A bayesian learning model in the agent-based bilateral negotiation between the coal producers and electric power generators. In *2008 International symposium on intelligent information technology application workshops* (pp. 859–862). IEEE.
60. Zhang, X., & Ma, H. (2018). Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations. arXiv preprint arXiv:1801.10459.
61. Zou, Y., Zhan, W., & Shao, Y. (2014). Evolution with reinforcement learning in negotiation. *PLoS ONE, 9*(7), e102840.