Reinforcement Learning for Automated Negotiation

Yasser Mohammad
NEC CORPORATION
Tokyo, Japan
y.mohammad@nec.com

ABSTRACT

Background: Developing effective autonomous negotiating agents that adapt to diverse scenarios and opponents remains a critical challenge for real-world applications. Most existing RL-based approaches fail to balance generalizability across scenarios with effective utilization of utility function structure.

Objectives and Research Questions: This work addresses how to design a codec (observation encoder and action decoder) that generalizes across negotiation scenarios while maintaining the most important information about outcome-space and utility function structure. We also investigate how to create a unified framework for representing and comparing different RL-based automated negotiation approaches.

Methods: We propose a vector utility-dependent outcome projection codec that projects outcomes onto multidimensional value function spaces. We develop a general framework representing negotiation problems as POMDPs and implement it using NegMAS and Gym. We evaluate the approach using Soft-Actor-Critic across multiple problem complexities.

Results: The proposed method outperforms state-of-the-art RL and heuristic methods in challenging scenarios with varying utility functions, outcome spaces, and opponents. It achieves statistically significant improvements in advantage, welfare, and fairness while maintaining competitive performance on simpler single-scenario problems.

Conclusions: The proposed codec successfully balances generalizability and information utilization without requiring feature engineering or constraining neural architectures. The unified framework enables systematic comparison of negotiation approaches and can represent most existing methods as special cases.

KEYWORDS

Automated Negotiation, Reinforcement Learning, Multi-Agent Systems

ACM Reference Format:
Yasser Mohammad. 2026. Reinforcement Learning for Automated Negotiation. In *Appears at the 8th Games, Agents, and Incentives Workshop (GAIW-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 19 pages.

Appears at the 8th Games, Agents, and Incentives Workshop (GAIW-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Armstrong, Curry, Hosseini, Mattei, Tsang, Wąs (Chairs), May 2026, Paphos, Cyprus.

1 INTRODUCTION AND RELATED WORK

The increasing prevalence of AI and multiagent systems in industrial and commercial operations has amplified the significance of coordinating self-interested agents within competitive environments. This has spurred considerable interest in automated negotiation as a key agreement technology for orchestrating agent interactions in such settings. A paradigmatic example of the use of automated negotiation in real-world business applications is the generalized procurement task in which a buyer and a seller negotiate over the price, quantity, item properties, delivery dates and related terms of a contract [37].Other applications of automated negotiation include permission management in IoT systems, Wi-Fi channel assignment [17, 28], agriculture supply chain support, supply chain management [33], vehicle routing [16], path planning [22], and providing feedback for student negotiation skills [23] [36] including procurement, supply chain management [33, 37], IoT permission management, and Wi-Fi channel assignment marsa2019nonlinear.

Automated negotiation has a rich research history, originating with the Nash Bargaining Game [39]. Early investigations focused on game-theoretic analyses [45], while recent research has shifted towards developing effective negotiation strategies [6, 7, 9, 15, 25, 51] and novel protocols [26, 32].

The Alternating Offers Protocol (AOP) [45] is the most commonly employed protocol for bilateral negotiations, while the Stacked Alternating Offers Protocol (SAOP) [6] extends it to multilateral scenarios. In SAOP, agents take turns offering complete agreement proposals; each agent can accept, reject and counter-offer, or withdraw. The process continues until agreement, withdrawal, or a predefined deadline is reached. Agents have utility functions mapping offers to values and reservation values for disagreement. This paper focuses on bilateral AOP (SAOP with two agents).

Initial efforts in strategy design centered on manually crafted heuristics [9, 25, 38, 51], with machine learning used for opponent modeling [8], parameter optimization [27], and algorithm configuration [12]. Heuristic strategies remain prevalent in the Automated Negotiating Agents Competition (ANAC) [4].

Reinforcement learning (RL) in negotiation was introduced a decade ago [41]. More recently, RL has been applied to offering strategies [2, 11], acceptance strategies [43], opponent modeling, strategy switching [46], end-to-end negotiation [21, 48], and concurrent negotiations [1, 10, 30]. Offline RL has also been explored [13, 14, 40].

Most existing RL methods employ two primary approaches for encoding negotiation history and decoding model actions (collectively called a *codec*), each with its own weaknesses: (1) The **Utility-Projection Codec**, where offers are represented by their utility for the agent, and optionally the opponent's utility if an opponent model

is available [2, 11, 43]. (2) The **Raw-Outcome Codec**, where offers are embedded directly without reference to the utility function [21, 48].

The Utility-Projection codec enables negotiation across diverse scenarios but sacrifices detailed offer information beyond utilities. For instance, in a buyer-seller negotiation involving price, quantity, and delivery date, if the seller consistently offers a specific delivery date due to production constraints, a buyer using Utility-Projection might not discern the significance of the delivery date and thus fail to adjust its offers, potentially leading to suboptimal results.

Conversely, the Raw-Outcome codec avoids this limitation because it allows the agent to observe the outcomes themselves instead of their utility function which provides more information (e.g., for opponent modeling). Nevertheless, this codec lacks generality. An agent trained on a specific trading scenario where the issues are quantity and price, for example, cannot be directly applied to a different scenario, such as employer-employee negotiations where the issues are the salary, paid leave days, and health insurance. Furthermore, even within the same domain, a change in utility functions can render a trained agent completely ineffective because the same input (i.e., same outcome) now represents a different trade-off in terms of the utility for itself and its opponent(s).

In summary, the Utility-Projection and Raw-Outcome codecs represent two extremes: the former offers full generality but discards valuable information, while the latter retains more information but lacks generalizability. It is important to note that the codec defines the *problem* being solved and is not learnable by the agent itself, and thus it is a design choice that can significantly affect the agent's practical value. Therefore, the choice of codec cannot be learned directly by the model.

A different approach was proposed by Renting et al. 2024 [44], who used a policy model based on Graph Neural Networks to deal with the changing dimensions of both the observation and action spaces. This allowed the system to adapt to different utility functions and outcome spaces. Our proposal is simpler, architecture-agnostic, and requires no feature engineering, while maintaining relevant information for the negotiation strategy. Unlike [44], our approach does not constrain the neural architecture and can work with any standard RL algorithm.

Contribution. The main contribution of this work is the introduction and evaluation of a Codec for encoding negotiation history and negotiator actions in the context of RL that avoids the pitfalls of Utility-Projection and Raw-Outcome codecs without using hand-coded features, constraining the neural architecture, or being only applicable to linear aggregation utility functions. As a secondary contribution, we introduce a general framework for representing RL for AN problems and a full modular implementation, which can represent most existing SOTA methods as well as a large variety of new approaches (NegMAS-RL). Our implementation builds upon the NegMAS [34] library for automated negotiation and Gym/Petting Zoo [49, 50] for RL/MARL. Finally, we utilize the proposed framework to evaluate the proposed codec against SOTA codecs and show improvements in agent advantage, welfare, and fairness for the most difficult RL for AN instances.

2 AUTOMATED NEGOTIATION (AN)

Formally, a negotiation *scenario* λ is defined as a tuple $(\mathcal{A}, \mathcal{D})$ where $\mathcal{A}$ is the set of agents (also called *negotiators*) numbered from 1 to n, and $\mathcal{D}$ is the negotiation domain. The negotiation domain $\mathcal{D} \equiv (\Omega, \mathcal{U})$ is pair: (1) The outcome space (Ω of size m) comprising all possible agreements. A special outcome $\phi \notin \Omega$ is always assumed to exist to represent disagreement and we define the *extended outcome space* Ω^+ as the $\Omega \bigcup \{\phi\}$. (2) $\mathcal{U}$ is a tuple of agent utility functions. Each agent utility function u_i is a mapping from Ω^+ to the range $[0, 1]$. The utility of disagreement ($u_i(\phi)$ is called the *reservation value*. Time-pressure can be modeled by a discounting factor as in [45] or by a limit on the number of rounds/seconds allowed for the negotiation [6]. We assume that agents know their own utility functions but not their opponents' utility functions.

The negotiation protocol $\mathcal{P}$ defines the actions available to the agents, when they can execute these actions, timeout conditions, and termination conditions. While applicable to a much wider set of negotiation protocols (i.e., all Generalized Bargaining Protocols defined in [32]) including multilateral protocols this paper focuses on the bilateral Alternating Offers Protocol (AOP) [6] described earlier to simplify the exposition.

A negotiation *strategy* $\pi_i\left(\mathcal{T}^t; n, i, \Omega, u_i\right)$ maps the negotiation history $\mathcal{T}^t$ to actions, where $\mathcal{T}^t = \left(A^0, A^1, \ldots, A^{t-1}_{-i}\right)$, and A^k represents all the actions from all agents on time step k. For AOP, $\mathcal{T}^t = \left(\omega_0(0), \omega_1(2), \ldots \omega_{k \pmod 2}(k), \ldots, \omega_{(t-1) \pmod 2}(t-1)\right)$ where

$\omega_{k \pmod 2}(k) \in \Omega^+ \cup \{\alpha\}$ with α representing agreement, ϕ representing leaving the negotiation. Actions of agents are sampled from the output of their strategies: $\omega_{k \pmod 2}(k) \sim \pi_{k \pmod 2}(\mathcal{T}^k)$.

Given a scenario λ, protocol $\mathcal{P}$ and strategy profile π, executing the negotiation yields the *negotiation outcome* $\omega^* \in \Omega^+$, denoted $\mathcal{P}(\lambda, \pi) = \omega^*$. Each agent receives utility $u_i(\omega^*)$, and the *advantage* is defined as $u_i(\omega^*) - u_i(\phi)$. We use $i \approx \pi$ to denote that agent i (assigned utility function u_i) uses strategy π.

So far, we did not assume any structure for the outcome space or the utility function. The outcome space Ω is usually defined as the Cartesian product of a set of n_I sets Ω_j called "issues" (i.e., $\Omega = \Omega_1 \times \Omega_2 \cdots \times \Omega_{n_I}$). For example, a trade negotiation can have three issues: price, quantity, and delivery date. The most widely used utility function structure for multi-issue negotiation is the *Linear Aggregation utility function* (LA) defined as:

$$u(\omega) = \sum_{j=1}^{n_I} \alpha_j \times \nu_j(\omega_j), \tag{1}$$

where $0 \le \alpha_j \le 1$, $\sum_{j=1}^{n_I} \alpha_j = 1$, $\nu_j : \Omega_j \to [0, 1]$ is the *value function* for issue j, and $\omega_j \in \Omega_j$ is the value of issue j in ω.

In this paper, we assume a generalization of this structure called "Generalized Linear Aggregation" utility functions (GLA) that has the form:

$$u(\omega) = \sum_{k=1}^{K} \alpha_k \times \nu_k(\omega_{g_k}), \tag{2}$$

where $0 \le \alpha_k \le 1$, $\sum_{k=1}^{K} \alpha_k = 1$, g_k is a subset of the issues called an issue-group, Ω_{g_k} is the Cartesian product of the issues in the

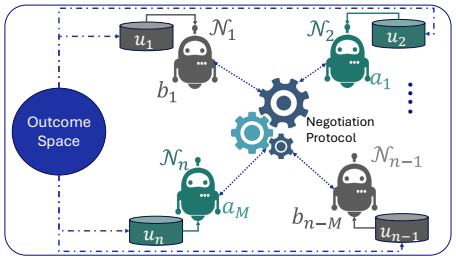


Figure 1: Learning to Negotiate Problem.

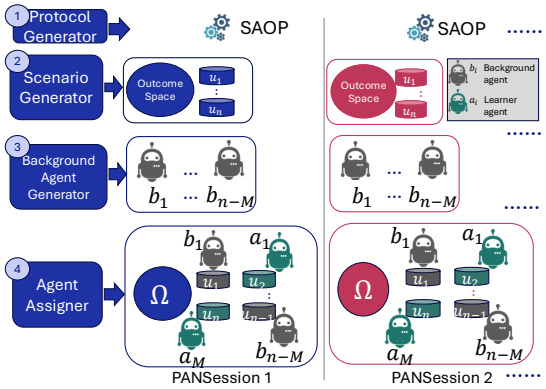


Figure 2: Proposed PANSession Generation Process (PGP).

issue-group, $v_k : \Omega_{g_k} \rightarrow [0, 1]$ is the *value function* for issue-group k, K is the number of issue-groups, and ω_{g_k} are the values of the issues belonging to the issue-group g_k in ω. For example, a car buyer may have issue-groups for "general condition" (mileage, accidents, paint) and "safety" (airbags, year, ABS), with some issues shared across groups.

2.1 The "RL for AN" Problem

To use RL/MARL for automated negotiation, we divide the agents during any negotiation into *learner* $\mathcal{A}$ and *background* $\mathcal{B}$ agents with only the former being trained using RL/MARL. *Background* agents can use pretrained models and change their *behavior* based on what happens during the negotiation but are considered as part of the environment for the RL/MARL algorithm.

An *M-learner Partially Assigned Negotiation Session (PANSession)* $\mathcal{S}$ =($\mathcal{P}$, λ, M, π_B, $\mathcal{I}$) defines a ready-to-run negotiation session consisting of a protocol and a negotiation scenario λ with M *learners* assigned to M out of the n agents and $n - M$ background strategies π_B assigned to the remaining agents. A *PANSession Distribution* $\mathcal{D}$ is a probability distribution over a set of PANSessions.

The *RL for AN Problem* (Fig. 1) is defined as: *Given two PANSession distributions called the training* $\mathcal{D}^{trn}$ *and testing* $\mathcal{D}^{tst}$ *distributions, train an agent on samples from* $\mathcal{D}^{trn}$ *using RL to maximize its expected advantage on* $\mathcal{D}^{tst}$.

3 THE PROPOSED RL FOR AN FRAMEWORK

One of the challenges motivating this paper was the need to define a general framework for representing different approaches to RL in automated negotiation. Until now, most RL approaches to automated negotiation have been developed independently with no common framework for representing the problem, making it difficult to compare and combine existing solutions. This section describes such a framework which consists of the following components mirroring the structure of the "RL for AN" problem defined above: (1) PANSession generation process which is responsible for modeling the training and testing distributions of negotiation sessions. (2) Modular components for representing the environment, state, action and reward functions. (3) A unified process for training, testing and deployment of RL agents based on the aforementioned components and generation process. The following sections describe these three components in turn. This framework is proposed both as a theoretical tool for characterizing RL for AN problems as well as an open-source library for developing solutions to these problems capable of representing most RL for AN methods available in the literature.

3.1 PANSession Generation Process

The goal of the PANSession Generation Process (PGP) is to sample Partially Assigned Negotiation Sessions from a PANSession distribution $\mathcal{D}$. We model $\mathcal{D} \equiv P(\mathcal{P})P(\Lambda|\mathcal{P})P(\mathcal{B}|\mathcal{P}, \lambda)P(\mathcal{I}|\mathcal{P}, \lambda, b)$ as the product of four distributions: a distribution over protocols $P(\mathcal{P})$, a distribution over scenarios conditioned on the sampled protocol $P(\Lambda|\mathcal{P})$, a distribution over background agent strategies conditioned on both the protocol and scenario sampled $P(\mathcal{B}|\mathcal{P}, \lambda)$, and a distribution over learner and background agent assignment $P(\mathcal{I}|\mathcal{P}, \lambda, b)$.

Fig. 2 details the process of generating a Partially Assigned Negotiation Session: (1) Firstly, the protocol is generated including any parameters such as time limits. (2) A scenario is sampled from the set of target negotiation scenarios. (3) $n - M$ background agents are generated. (4) M placeholders are assigned to the learner agents each connecting a learner's policy to its negotiating agent. Background strategies are assigned to the remaining $n - M$ agents. This process leads to a ready-to-run negotiation session to be used for training or testing the learners.

3.2 Components

An RL for AN problem is represented by a *NegoEnv*/*NegoMultiEnv* environment which runs the PGP described above to sample PANSessions as needed for training and testing the learner/learners respectively. The top of Fig. 3 shows this process.

A PANSession generated through the PGP represents the initial state distribution and the state transition model (environment) for the learner RL policy. To simplify the exposition we will assume a single learner agent in this section, extension to multiple agents is straightforward and supported by the proposed framework through *NegoMultiEnv*.

To define an RL problem, we need to define POMDP components (state space, transition model, action space, observation model, reward function). For RL for AN, this is challenging: (1) The protocol, background agent utilities and strategies all contribute to

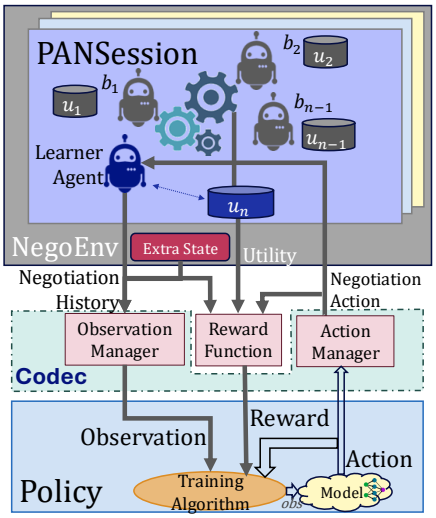


Figure 3: Components of the proposed framework.

environment dynamics; (2) Markovian states must represent the entire (arbitrarily long) history, and the representation depends on the outcome space; (3) action and state spaces are episode-dependent as they vary with outcome spaces. These issues make all aspects episode-dependent except in the simplest case of training and testing in a single scenario with a fixed opponent. Additionally, the natural reward (utility at agreement) is sparse, delayed, and episode-dependent.

An RL for AN training problem is represented by the tuple $(\mathcal{D}^{trn}, C \equiv (\mathcal{OE}, \mathcal{AD}), \mathcal{RF})$ which maps to a finite-horizon POMDP $(\mathcal{S}_m, \mathcal{A}_m, O_m, \mathcal{T}_m, \mathcal{R}_m)$ where $\mathcal{D}^{trn}$ is the training PANSession distribution, C is the codec consisting of an observation encoder $\mathcal{OE}$ and an action decoder $\mathcal{AD}$, and $\mathcal{RF}$ is the reward function. For the resulting POMDP, $\mathcal{S}_m$ is the state space, $\mathcal{A}_m$ is the action space defined by $\mathcal{AD}$, O_m is the observation space defined by $\mathcal{OE}$, $\mathcal{T}_m$ is the transition model defined by the PGP, the negotiation protocol and the background agents, and $\mathcal{R}_m$ is the reward model defined by $\mathcal{RF}$. The goal of training is to find a policy $\pi : O_m \rightarrow p[\mathcal{A}_m]$ that maximizes the expected cumulative reward over episodes sampled from $\mathcal{D}^{trn}$, where $p[X]$ is a probability distribution over X. Testing is done similarly using a testing distribution $\mathcal{D}^{tst}$.

The **Observation Encoder** $\mathcal{OE}$ presents the state of the negotiation (and any external contextual state) to the learner's policy defining the state-space. The most widely used approach is to use a moving window of some predefined length L over opponent offers and then apply utility projection (U) either to discrete bins (Ud) or a continuous real range (Uc) to each outcome creating an L-dimensional vector (i.e., $\mathcal{S}_m = [0,1]^L$). Bakker et al. 2019 [11] used the difference from the last offered utility instead of an absolute utility value (Udr). These approaches have the advantage of applicability to any outcome space but, as discussed earlier, are incapable of utilizing the internal structure of the outcome space or utility function. Using raw outcomes is also common either using one-hot encoding over all outcomes (Od) [48] (i.e., $\mathcal{S}_m = \{0,1\}^{|\Omega| \times L}$) or by encoding every issue independently in a continuous or discrete space with one-hot encoding being the most widely used (Oid/Oic) [21] (i.e., $\mathcal{S}_m = \prod \{0,1\}^{|\Omega_j| \times L}$ where Ω_j is the domain of issue J). These approaches suffer from being applicable only to a single scenario (i.e., $P(\Lambda) = \delta[\lambda = \hat{\lambda}]$ for some specific scenario $\hat{\lambda}$). Any change in the outcome space or utility functions will require retraining [47].

The **Action Decoder** $\mathcal{AD}$ converts the output of the learner's policy into a valid negotiation action defining the action-space. By using an appropriate action decoder, it is possible to model offering policies [46], end-to-end approaches [21, 47]), acceptance policies [43], and strategies utilizing an external opponent model [11]. Systems learning only an acceptance strategy will have a binary or single-dimensional continuous action space (A) leading to $\mathcal{A}_m = \{0,1\}$. Systems learning an offering policy will have the same options as the observation encoder with the same kinds of limitations and advantages (with $L = 1$). End-to-end systems usually combine these two action spaces through concatenation leading to a Cartesian product of the action spaces. The action decoder is usually the inverse of the observation encoder (i.e., both use or both do not use utility projection) but the proposed framework makes it possible to combine any observation encoder with any action decoder to create a new codec.

The **Reward Function** $\mathcal{RF}$ defines the reward received by the learner's policy after each transition of the environment (i.e., every step of the negotiation protocol execution). The intrinsic reward of the RL for AN problem is the advantage received at the end of the negotiation which can range between -1 and 1. Nevertheless, it is common practice to use some form of reward shaping (e.g. a negative reward for disagreement [10, 43] or normalization of the utility by opponent's utility [2]).

Framing the RL for AN problem in this proposed framework immediately suggests new codecs by combining different observation encoders and action decoders, and using offers from the agent itself, not only the opponents, in the observation.

3.3 Modeling SOTA RL methods in the proposed framework

The proposed framework can represent a wide variety of negotiation problems as well as RL solutions to them. Table 1 describes how to define several of the state of the art RL methods in terms of the observation and action encoders, and reward shaping as well as the ability to utilize information about the internal structure of the utility function. The paper defines the different options for the observation and action encoders shown in the table.

3.4 Training/Testing and Deployment Process

Fig. 3 shows the unified training process, which is agnostic to policy representation and training algorithms, enabling it to model almost all existing RL for AN systems. During training, NegoEnv samples a PANSession using the PGP (Section 3.1) and runs the negotiation protocol. At each step, the observation encoder represents the state, the reward function generates rewards, and the action decoder converts policy outputs to valid negotiation actions (counter-offer, acceptance, or withdrawal). This process repeats until the negotiation completes and the negotiation outcome is known. Testing uses the trained model without rewards; the observation encoder and action

Table 1: Modeling existing RL for AN methods in terms of the proposed framework. "Reward Shaping" is + if reward shaping is used. "Can Generalize" is + if the scenarios used for training and testing can differ and "Uses OS Structure" is + if the agent can utilize the internal outcome-space and utility function structure.

Method	Observation Encoder	Action Decoder	Reward Shaping	Can Generalize	Uses OS Structure	No Feature Engineering
Razeghi et al. 2020 [43]^{aM}	Ud	A	+	+	-	+
Matsuo and Fujita 2024 [29]^{a}	Uc	A		+	-	+
Bakker et al. 2019 [11]^{oM}	Udr	Udr		+	-	+
Sengupta et al. 2021 [46]^{o}	Uc	Uc		+	-	+
Bagga et al. 2021 [10]^{e†}	Uc	Uc	+	+	-	+
Sengupta et al. 2022 [47]^{e}	Uc	Uc		+	-	+
Arslan and Aydoğan 2022 [2]^{e}	Uc	Uc	+	+	-	+
Miyajima and Fujita 2024 [30]^{o†}	Uc	Uc	+	+	-	+
Chen et al. 2024 [14]^{e‡}	Ic	Ic		-	+	+
Arakawa and Fujita 2023 [1]^{e†}	Id	Id	+	-	+	+
Chen et al. 2023 [13]^{e‡}	Ic	Ic		-	+	+
Higa et al. 2023 [21]^{e}	Id	Id		-	+	+
Takahashi et al. 2022 [48]^{o}	Od	Od		-	+	+
Renting et al. 2024 [44]^{e}	Gd	Gd		+	+	-
VUDO (Proposed)	VUDO	VUDO	±	+	+	+

^{e} End-to-End ‡ Offline RL † Concurrent
^{o} Offering ^{a} Accepting ^{M} Requires an Opponent Model
Gd: Graph-discrete (using Graph Neural Networks)

decoder are included with the model for deployment. A full implementation based on NegMAS [35] is provided as an open-source library; all experiments in this paper use this framework.

4 THE PROPOSED CODEC: VUDO

As discussed earlier almost all existing approaches (with a single exception [44]) use either utility projection with its low information utilization or the Raw-Outcome Codec with its low generalizability. Two problems face the Raw-Outcome Codec: (1) different negotiation scenarios will have different action and state space sizes and dimensions which is incompatible with traditional neural architectures. This is what [44] solves using a Graph Neural Network (GNN) that utilizes hand-crafted features to unify the representation size of observations and actions. (2) the meaning of an offer from the opponent can change completely if the utility function of the opponent (or their model of the agent's own utility function) changes even without any change in the state and action spaces.

We start by noticing that the agent's objective is *always* to maximize its expected advantage. This suggests that the raw-values of outcomes are of no consequence except in how they relate to the utility functions of the two agents. Consider the special case when every outcome of a GLA (Eq. 2) has a unique combination of values for all agents of a negotiation. Under this condition, knowing this combination of values (and the corresponding weights) captures *all* information about this outcome as the outcome can be recovered from these values (and the weights). Now assume that two outcomes map to the same combination of values for all agents. Even though it is not possible to recover the outcome by knowing this combination of values, the two outcomes must, by definition, have the same utility for all agents and are in the same relative rank in every utility function with the same relationship (in terms of utility) to all other outcomes. The main principle behind the work reported in this paper is that *for the sake of negotiation performance*, these two outcomes are *equivalent* and the difference between them is not important for the negotiating agent. This is not advanced as a provable theoretical claim but as a design principle.

Based on this design principle, we propose projecting outcomes onto the multidimensional space defined by the value functions of a GLA utility function. Computationally, this amounts to a vector embedding of outcomes in a utility dependent space (Vector Utility-Dependent Outcome projection, VUDO).

Algorithm 1 shows this process for the observation encoder for a target dimensionality of d. Line 2 shows the basic projection step. Note that instead of projecting the outcome ω into $2 \times K$ weight and value dimensions $(\alpha_k, v_k(\omega_{g_k}))$, we project it into the K weighted-value dimensions. In practice, we did not notice any empirical difference between the two approaches in terms of the quality of trained models. If the GLA has too few value functions, we simply divide the one with highest weight recursively until we reach the target dimensionality (Algorithm 1, lines 3-9). If the GLA has too many value functions, we combine the ones with lowest weights recursively until the target dimensionality is reached (Algorithm 1, lines 9 to 17).

The action decoder needs to convert the VUDO projected vector to an outcome from the outcome space. This is an inverse problem which means it will be computationally more intensive than the forward VUDO projection. In this paper, we use a simple approach. First, we project the entire outcome space using Algorithm 1 and

Algorithm 1 VUDO Observation Projection for GLA Ufuns

function VUDOO($\omega \in \Omega; d, g_k, \alpha_k, \nu_k \forall k \in [1, K]$)
  $x \leftarrow \left\langle \alpha_k \nu_k(\omega_{g_k}) \forall 1 \le k \le K \right\rangle$
  while $K < d$ **do**
    $m \leftarrow \text{argmax}_{1 \le k \le K} \alpha_k$
    $x_m \leftarrow \alpha_m \nu_m(\omega_{g_m})/2$
    $x_{K+1} \leftarrow x_m$
    $\alpha_k \leftarrow \left\langle \begin{cases} \alpha_k & k \notin \{m, K+1\} \\ \alpha_m/2 & \text{otherwise} \end{cases} \right\rangle$
    $K \leftarrow K + 1$
  while $K > d$ **do**
    $n_1 \leftarrow \text{argmin}_{1 \le k \le K} \alpha_k$
    $n_2 \leftarrow \text{argmin}_{1 \le k \le K, k \ne n_1} \alpha_k$
    $x_{n_1} \leftarrow \alpha_{n_1} \nu_{n_1}(\omega_{g_{n_1}}) + \alpha_{n_2} \nu_{n_2}(\omega_{g_{n_2}})$
    $x \leftarrow \langle x_k, k \in [1, K] \wedge k \ne n_2 \rangle$
    $\alpha_{n_1} \leftarrow \alpha_{n_1} + \alpha_{n_2}$
    $\alpha \leftarrow \langle \alpha_k, k \in [1, K] \wedge k \ne n_2 \rangle$
    $K \leftarrow K - 1$
  return x

cache the results. This projection is only performed once per scenario. Then, to find an outcome for a given vector x, we find the nearest projected outcome in the cache using a predefined distance metric $D(.,.)$. We use cosine similarity in all experiments in this paper. While this approach does not scale well to very large outcome spaces, any approximate method for projection inversion (e.g., approximate nearest neighbor search) can be used instead.

5 EVALUATION

We conducted a series of experiments to evaluate the proposed approach against SOTA negotiation strategies in situations with different complexity (i.e., different PANSession distribution widths)[1]. To have a fair comparison, all methods were trained using Soft-Actor-Critic (SAC) [19] which achieves best results in earlier studies [2, 21, 43, 46, 47] except Renting$^+$ [44] for which we used PPO because it is recommended by the authors. Stable-Baselines3's SAC implementation [42] was used for all experiments with default hyperparameters. We also evaluated Proximal Policy Optimization (PPO) but achieved higher advantage using SAC for all algorithms except Renting$^+$. NegMAS [34] was used for all experiments and all negotiations were limited to 100 rounds. All methods encode relative time (normalized to $[0, 1]$) as part of the observation. The proposed framework was used to implement all RL agents except Renting$^+$ for which we used the official implementation [44]. We used a simple MLP neural architecture for all architecture-independent methods.

We used the following SOTA RL agents as baselines:

RLBOA$^+$ [11] Learns an offering strategy using the Utility-Projection codec. It uses an opponent model for selecting final offers (implemented in the action decoder). The original algorithm used Q-learning but [21] showed that SAC achieves better results on the same architecture. In this paper, we provide RLBOA with the *exact* utility function of the opponent making its performance an upper limit independent of the opponent model used.

Sengupta [47] An End-to-End learner using utility projection.

VeNAS [48] Learns an offering strategy using the Raw-Outcome codec.

MiPN [21] An End-to-End learner using raw outcomes.

Renting$^+$ [44] An End-to-End learner using a GNN to encode outcomes based on 9 predefined features. To avoid changes in behavior that may be introduced due to reimplementation using different library versions under NegMAS-RL, we used the official implementation of this algorithm [44] which used GeniusWeb [18] as the negotiation platform. For this model, the training and testing opponent was always the Boulware time-based negotiation strategy which concedes slowly over negotiation time. This makes the reported performance an upper bound.

We also tested two ablation variations of VUDO: VUDONoAct uses VUDO only in for the observation encoder and VUDONoObs uses it only for the action decoder.

5.1 Evaluation Metrics

The main evaluation criterion in this paper is the expected **Advantage** on the test PANSessiondistribution (i.e. expected utility of negotiation outcome subtracted from the reservation value), an agent receives from employing a strategy π [31].

Formally, **Advantage** is defined for a PANSession test distribution $\mathcal{D}$ as:
$A_\pi(\mathcal{D}) \equiv \mathbb{E}_{\mathcal{P},\lambda,\pi^{-1},i\sim\mathcal{D}|\exists i, i\approx\pi}\left[u_i(\mathcal{P}(\lambda,\pi)) - u_i(\phi)\right]$,
where $\mathcal{P}$ is the sampled protocol (Always Stacked Alternating Offers in these experiments), λ is the sampled scenario in which agent i uses strategy π, π^{-1} are background agents and i is the index of the evaluated strategy in the sampled scenario (i.e. its utility function index).

We also use the following additional evaluation metrics as recommended by Mohammad 2023a [31]:

Welfare W is defined as the expected sum of the achieved value for all agents relative to the maximum achievable sum:
$W_\pi(\mathcal{D}) = \mathop{\mathbb{E}}_{\mathcal{P},\lambda,\pi^{-1},i\sim\mathcal{D}|\exists i, i\approx\pi}\left[\sum_{k\in\mathcal{A}} u_k(\mathcal{P}(\lambda,\pi))/\max_{\omega\in\mathbf{P}} \sum_{k\in\mathcal{A}} u_k(\omega)\right]$,
where $\mathbf{P}$ is the Pareto Outcome Set of the negotiation scenario:

$$\mathbf{P} = \left\{\omega \in \Omega : \nexists \psi \in \Omega \text{ s.t. } (\forall i \in \mathcal{A}, u_i(\psi) \ge u_i(\omega)) \wedge \left(\exists i' \in \mathcal{A}, u_{i'}(\psi) > u_{i'}(\omega)\right)\right\}$$

Pareto Optimality O is defined as the expected value of one minus the normalized distance of the negotiation outcomes to the rational portion of the Pareto Outcome Set for negotiations leading to agreement:
$O = \mathbb{E}_{\mathcal{P},\lambda,\pi^{-1},i\sim\mathcal{D}|\exists i, i\approx\pi}\left[1 - \min_{\omega\in\mathbf{P}\bigcup\overline{\Omega}} D(\mathcal{P}(\lambda,\pi),\omega)\right]$,
where $\overline{\Omega} = \{\omega \in \Omega s.t. \exists i \in \mathcal{A}, u_i\omega \ge u_i\phi\}$
and $D(a,b) \equiv \frac{\sqrt{\sum_{i\in\mathcal{A}}(u_i(a)-u_i(b))^2}}{\max_{\omega\in\Omega\wedge\psi\in\mathbf{P}\bigcup\overline{\Omega}} \sqrt{\sum_{i\in\mathcal{A}}(u_i(\omega)-u_i(\psi))^2}}$.

[1]The technical appendix provides full details of all datasets, experiments, hypothesis testing results and more evaluation metrics.

Table 2: Results of the Generalized Procurement Task (Section 5.2).

Method	Advantage	Welfare	Fairness (Nash)
Atlas3	0.154 (0.279)	0.297 (0.375)	0.425 (0.284)
Hardheaded	0.176 (0.354)	0.240 (0.307)	0.340 (0.179)
AgentK	0.201 (0.380)	0.259 (0.299)	0.349 (0.178)
AgentGG	0.174 (0.348)	0.238 (0.298)	0.345 (0.174)
Sengupta	0.191 (0.328)	0.323 (0.379)	0.448 (0.292)
RLBOA$^+$	0.160 (0.306)	0.279 (0.349)	0.389 (0.251)
Renting$^+$	0.230 (0.253)	0.378 (0.277)	0.475 (0.214
VUDO	**0.298 (0.272)**	**0.683 (0.329)**	**0.678 (0.239)**
VUDONoAct	0.215 (0.319)	0.376 (0.407)	0.497 (0.325)
VUDONoObs	0.223 (0.342)	0.417 (0.393)	0.467 (0.254)

Table 3: Results on ANAC 2024 scenarios (Section 5.3).

Method	Advantage	Welfare	Fairness (Nash)
Atlas3	0.163 (0.300)	0.222 (0.398)	0.522 (0.209)
AgentK	0.153 (0.350)	0.158 (0.362)	0.454 (0.140)
AgentGG	0.040 (0.196)	0.040 (0.196)	0.448 (0.133)
Hardheaded	0.040 (0.196)	0.040 (0.196)	0.448 (0.133)
RLBOA$^+$	0.077 (0.141)	0.212 (0.367)	0.504 (0.186)
Sengupta	0.117 (0.251)	0.887 (0.123)	0.529 (0.149)
Renting$^+$	0.018 (0.120)	0.451 (0.246)	**0.635 (0.204)**
VUDO	**0.179 (0.323)**	0.227 (0.406)	0.511 (0.196)
VUDONoAct	0.048 (0.111)	0.334 (0.429)	0.495 (0.155)
VUDONoObs	0.050 (0.016)	**0.891 (0.123)**	0.481 (0.113)

Table 4: Results on selected ANAC scenarios (Section 5.4).

Method	Advantage	Welfare	Fairness (Nash)
AgentGG	0.800 (0.422)	0.699 (0.377)	0.605 (0.255)
AgentK	**0.947 (0.049)**	**0.910 (0.080)**	**0.795 (0.141)**
Atlas3	0.684 (0.130)	0.868 (0.091)	0.855 (0.077)
Hardheaded	0.800 (0.422)	0.699 (0.377)	0.605 (0.255)
VeNAS(c)	0.541 (0.372)	0.706 (0.457)	0.694 (0.327)
MiPN(c)	0.475 (0.132)	0.860 (0.070)	0.733 (0.080)
RLBOA$^+$	0.846 (0.346)	0.782 (0.333)	0.702 (0.234)
Sengupta	0.828 (0.304)	0.831 (0.255)	0.723 (0.216)
Renting$^+$	0.636 (0.385)	0.588 (0.356)	0.539 (0.231)
VUDO	0.915 (0.175)	0.867 (0.107)	0.727 (0.154)
VUDO-NoAct	0.856 (0.209)	0.876 (0.149)	0.751 (0.197)
VUDO-NoObs	0.824 (0.340)	0.863 (0.108)	0.707 (0.164)

The reason we only consider the *rational* portion of the Pareto Outcome Set is to penalize negotiations ending with Pareto efficient irrational agreements.

Fairness F is defined as the expected value of one minus the normalized distance of the negotiation outcomes to any bargaining solution:
$F = \mathbb{E}_{\mathcal{P},\lambda,\pi^{-1},i\sim\mathcal{D}|\exists i,i\approx\pi}\left[1 - \min_{\omega\in\Omega^f} D(\mathcal{P}(\lambda,\pi),\phi)\right]$,
where Ω^f are the set of bargaining solutions under the single-negotiation and multiple-negotiations model (See the following section for the rationale for this definition and the details of the bargaining solutions set). We use the following two bargaining solutions to measure fairness:

Nash Bargaining Solution Ω_n the unique Nash Equilibrium [39] for the Nash Bargaining Game that satisfies Perto-optimality, symmetry, scale-invariance and independence of irrelevant alternatives (IIA):

$$\Omega_n \equiv \underset{\omega\in\mathbf{P}}{\operatorname{argmax}} \prod_{i\in\mathcal{A}} u_i(\omega) - u_i(\phi)$$

Kalai Bargaining Solution Ω_k Defined by [24] as the unique Nash Equilibrium when scale-invariance is dropped and IIA and resource monotonicity axioms are kept:

$$\Omega_k^* \equiv \underset{\omega\in\mathbf{P}}{\operatorname{argmax}} \min_{i\in\mathcal{A}} u_i(\omega) - u_i(\phi)$$

5.2 Generalized Procurement Task (Hard)

The base scenario is inspired by the procurement problem from the Supply Chain Management League (SCML) [36, 37]. The negotiation simulates a seller and buyer with issues: price, quantity, and optionally carbon-footprint. Utility functions were sampled randomly with average opposition level 0.37 (measuring difficulty of finding mutually beneficial agreements as normalized distance from Pareto frontier to ideal point; 0=identical utilities, 1=zero-sum).

Training opponents: Atlas3 [38], CUHK [20], Boulware, Conceder. Testing opponents: AgentK [25], Hardheaded [51], AgentGG [5], Linear. This is the most challenging case: utility functions, issues, reservation values, and opponents all vary between training and testing. Raw-Outcome methods cannot be applied here because the outcome space changes between episodes. We trained for $400,000$ steps and tested on 50 negotiations.

Table 2 shows results. The proposed method achieves highest advantage, welfare, and fairness among all methods (differences statistically significant via Wilcoxon test with Bonferroni correction).

Moreover, both ablation variants achieved lower advantage than full VUDO (statistically significant, Wilcoxon $p < 0.05$ with Bonferroni correction), showing the codec benefits both observation encoding and action decoding.

Model sizes were similar: Sengupta $(341,766)$, VUDO $(354,060$, +3.5%), Renting$^+$ $(467,460)$. Training times: Sengupta (2h29m), VUDO (2h31m, +1.5%), Renting$^+$ (13h23m). VUDO was fastest at negotiation (0.112 sec/round).

5.3 ANAC 2024 Scenarios (Hard)

We used 100 training and 50 testing scenarios from ANAC 2024 [3] with the same opponents and parameters as Section 5.2. These scenarios represent real-world negotiation domains from the competition. Table 3 shows results. The proposed method outperformed all others in *advantage* (statistically significant except vs. Sengupta and AgentK), though not in welfare/fairness. The improvement over

Table 5: Results on Selected ANAC Scenarios with scenario generalization (Section 5.5).

Scenario	AgentGG	AgentK	Atlas3	Hardheaded	RLBOA$^+$	Sengupta	Renting$^+$	VUDO
Advantage	0.720 (0.449)	**0.959 (0.042)**	0.668 (0.115)	0.720 (0.449)	0.776 (0.347)	0.726 (0.265)	0.478 (0.255)	0.893 (0.163)
Welfare	0.611 (0.390)	**0.892 (0.074)**	0.870 (0.100)	0.611 (0.390)	0.756 (0.322)	0.853 (0.088)	0.486 (0.348)	0.878 (0.101)
Nash	0.528 (0.246)	0.746 (0.123)	**0.866 (0.082)**	0.528 (0.246)	0.669 (0.210)	0.673 (0.143)	0.502 (0.192)	0.735 (0.142)

ablation variants confirms the codec is useful for both input and output.

5.4 Single Scenario, Single Opponent (Easy)

For this experiment, we used the same scenario for training and testing. Training and testing were conducted using the Boulware time-based negotiation strategy. This is the simplest possible negotiation situation. We used 200,000 training steps and tested all strategies using 30 negotiations on each scenario. This is the only experiment in which we could apply Raw-Outcome methods because the training and testing scenarios were the same.

We used 8 scenarios from the ANAC competition (Acquisition, Laptop, Camera, Car, Itex vs. Cypress, Thompson, EnergySmallA, Grocery) selected to have a wide variety of outcome space sizes (27 to $15,625$) and opposition levels (0.092 to 0.430)[2].

Table 4 shows the results of this experiment. The proposed method outperformed all other RL methods and SOTA heuristics except AgentK in terms of advantage. AgentK achieved a slightly higher average advantage on these scenarios but the difference was *not* statistically significant according to a Wilcoxon signed-rank test. Again variations of VUDO achieve lower scores which supports the value of the proposed codec in both observation encoding and action decoding.

5.5 Scenario Generalization (Medium)

In this experiment, we use the selected ANAC scenarios used in the previous experiment (Section 5.4). This time, we train using 5 scenarios and test on the remaining 3. We used the same training and testing opponents as in Section 5.2. This experiment is of medium difficulty as the agent needs to learn to generalize over scenarios but not over opponent strategies.

Table 5 shows the results of this experiment comparing the proposed method with SOTA heuristics and SOTA RL methods. The proposed method outperforms all comparison methods in terms of advantage except AgentK. Table 6 shows the ablation study for this scenario set. Again, the proposed method outperformed both variations in all metrics except negotiation time suggesting that the proposed codec is useful for both encoding observations and decoding actions.

5.6 Limitations and Future Directions

Taken together, these results show that VUDO outperforms all other RL methods in all experiments. Moreover, it achieves similar performance to SOTA heuristic methods in simple situations where training and testing are done on the same scenario against the same opponent, while generalizing better to harder situations with varying utility functions, outcome spaces, reservation values, and opponent strategies. Comparing VUDO to its ablation variants shows that both observation encoding and action decoding contribute to this performance gain. The approach extends to opponent modeling [8] and supervised learning [40].

[2]We employed all the scenarios used by the authors of VeNAS [48], MiPN [21], and Sengupta [46] in their evaluations. See the technical appendix for details.

Table 6: Ablation study for Scenario Generalization (Section 5.5).

Scenario	VUDO	VUDONoAct	VUDONoObs
Advantage	**0.893 (0.163)**	0.841 (0.260)	0.780 (0.324)
Welfare	**0.878 (0.101)**	0.841 (0.199)	0.782 (0.308)
Nash	**0.735 (0.142)**	0.709 (0.182)	0.698 (0.218)

The main limitation is the assumption of GLA utility functions (Eq. 2). In real-world scenarios, the utility function may only be available as a black-box with no ability to define clear value functions of issue-groups. In such cases, VUDO reduces to Utility-Projection. An auto-encoder could learn vectorized utility representations, which would also address scalability concerns from the nearest-neighbour search in action decoding. While data structures like k-d trees and locality-sensitive hashing can ameliorate the scalability problem, the auto-encoder approach provides a more principled solution.

Future work includes exploring advanced architectures (Transformers, RNNs) that exploit negotiation history and the natural permutation symmetries in VUDO—applying any consistent permutation to the input should not affect the policy since this amounts to permuting issues, which should not affect agent decisions. This symmetry could be implemented using, for example, a 1×1 convolution first layer. While our framework supports multiagent RL and multilateral protocols, we focused on bilateral negotiation. The last two experiments show that RL methods (including VUDO) do not consistently outperform SOTA heuristics, suggesting room for algorithmic and architectural improvement including MARL techniques.

6 CONCLUSION

We presented a new codec of outcomes for observation encoding and action decoding in automated negotiation and showed empirically that it can improve the performance of reinforcement learning on this problem outperforming SOTA RL and heuristic methods when generalization over negotiation scenarios and opponent strategies is required. The proposed VUDO codec projects outcomes onto the multidimensional space defined by value functions, capturing all information relevant to negotiation performance while enabling generalization across scenarios. Moreover, we presented

a general framework for representing automated negotiation problems as POMDPs for RL solvers. The proposed model is modular, open-source and can represent most available methods and can be used to develop new approaches and architectures. We hope that the introduction of this framework and the proposed codec can provide a solid foundation for future research in this area.

REFERENCES

[1] Ryota Arakawa and Katsuhide Fujita. 2023. Deep Deterministic Policy Gradient for Nested Parallel Negotiation. In *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 197–204.

[2] Furkan Arslan and Reyhan Aydoğan. 2022. Actor-critic reinforcement learning for bidding in bilateral negotiation. *Turkish Journal of Electrical Engineering and Computer Sciences* 30, 5 (2022), 1695–1714.

[3] Reyhan Aydoğan, Tim Baarslag, Tamara CP Florijn, Katsuhide Fujita, Catholijn M Jonker, and Yasser Mohammad. 2025. [COMP24] The Automated Negotiating Agents Competition (ANAC) 2024 Challenges and Results. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 3000–3002.

[4] Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Holger H Hoos, Catholijn M Jonker, Yasser Mohammad, and Bram M Renting. 2022. The 13th international automated negotiating agent competition challenges and results. In *International Joint Conference on Artificial Intelligence*. Springer, 87–101.

[5] Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Johnathan Mell, Jonathan Gratch, Dave De Jonge, Yasser Mohammad, Shinji Nakadai, Satoshi Morinaga, Hirotaka Osawa, et al. 2020. Challenges and main results of the automated negotiating agents competition (ANAC) 2019. In *Multi-Agent Systems and Agreement Technologies: 17th European Conference, EUMAS 2020, and 7th International Conference, AT 2020, Thessaloniki, Greece, September 14-15, 2020, Revised Selected Papers 17*. Springer, 366–381.

[6] 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.

[7] Tim Baarslag, Enrico H Gerding, Reyhan Aydogan, and MC Schraefel. 2015. Optimal negotiation decision functions in time-sensitive domains. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Vol. 2. IEEE, 190–197.

[8] Tim Baarslag, Mark JC Hendrikx, Koen V Hindriks, and Catholijn M Jonker. 2016. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems* 30, 5 (2016), 849–898.

[9] 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.

[10] Pallavi Bagga, Nicola Paoletti, Bedour Alrayes, and Kostas Stathis. 2021. A deep reinforcement learning approach to concurrent bilateral negotiation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (Yokohama, Yokohama, Japan) *(IJCAI'20)*. Article 42, 7 pages.

[11] Jasper Bakker, Aron Hammond, Daan Bloembergen, and Tim Baarslag. 2019. RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents.. In *AAMAS*. 260–268.

[12] CM Jonker BM Renting, HH Hoos. 2020. Automated Configuration of Negotiation Strategies. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1116–1124.

[13] Siqi Chen, Jianing Zhao, Gerhard Weiss, Ran Su, and Kaiyou Lei. 2023. An effective negotiating agent framework based on deep offline reinforcement learning. In *Uncertainty in Artificial Intelligence*. PMLR, 324–335.

[14] Siqi Chen, Jianing Zhao, Kai Zhao, Gerhard Weiss, Fengyun Zhang, Ran Su, Yang Dong, Daqian Li, and Kaiyou Lei. 2024. ANOTO: Improving Automated Negotiation via Offline-to-Online Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2195–2197.

[15] Dave de Jonge. 2022. An Analysis of the Linear Bilateral ANAC Domains Using the MiCRO Benchmark Strategy. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*.

[16] Dave de Jonge, Filippo Bistaffa, and Jordi Levy. 2022. Multi-objective vehicle routing with automated negotiation. *Applied Intelligence* (2022), 1–24.

[17] Enrique De La Hoz, Ivan Marsa-Maestre, Jose Manuel Gimenez-Guzman, David Orden, and Mark Klein. 2017. Multi-agent nonlinear negotiation for Wi-Fi channel assignment. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1035–1043.

[18] GeniusWeb. 2025. *GeniusWeb: https://ii.tudelft.nl/GeniusWeb/*. https://ii.tudelft.nl/GeniusWeb/

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[20] Jianye Hao and Ho-fung Leung. 2014. *CUHKAgent: An Adaptive Negotiation Strategy for Bilateral Negotiations over Multiple Items*. Springer Japan, 171–179.

[21] Ryota Higa, Katsuhide Fujita, Toki Takahashi, Takumu Shimizu, and Shinji Nakadai. 2023. Reward-based negotiating agent strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11569–11577.

[22] Hiroaki Inotsume, Aayush Aggarwal, Ryota Higa, and Shinji Nakadai. 2020. Path negotiation for self-interested multirobot vehicles in shared space. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11587–11594.

[23] Emmanuel Johnson, Jonathan Gratch, and David default. 2017. Towards An Autonomous Agent that Provides Automated Feedback on Students' Negotiation Skills. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 410–418.

[24] Ehud Kalai. 1977. Proportional solutions to bargaining situations: interpersonal utility comparisons. *Econometrica: Journal of the Econometric Society* (1977), 1623–1630.

[25] Shogo Kawaguchi, Katsuhide Fujita, and Takayuki Ito. 2013. *AgentK2: Compromising Strategy Based on Estimated Maximum Utility for Automated Negotiating Agents*. Springer Berlin Heidelberg, Berlin, Heidelberg, 235–241.

[26] Mark Klein, Peyman Faratin, Hiroki Sayama, and Yaneer Bar-Yam. 2003. Negotiating complex contracts. *Group Decision and Negotiation* 12, 2 (2003), 111–125.

[27] Raymond YK Lau, Maolin Tang, On Wong, Stephen W Milliner, and Yi-Ping Phoebe Chen. 2006. An evolutionary learning approach for adaptive negotiation agents. *International journal of intelligent systems* 21, 1 (2006), 41–72.

[28] Ivan Marsa-Maestre, Enrique de la Hoz, Jose Manuel Gimenez-Guzman, David Orden, and Mark Klein. 2019. Nonlinear negotiation approaches for complex-network optimization: a study inspired by Wi-Fi channel assignment. *Group Decision and Negotiation* 28, 1 (2019), 175–196.

[29] Hyuga Matsuo and Katsuhide Fujita. 2024. Effective Acceptance Strategy Using Deep Reinforcement Learning in Bilateral Multi-issue Negotiation. *Journal of Information Processing* 32 (2024), 2–9.

[30] Ryoga Miyajima and Katsuhide Fujita. 2024. Deep Reinforcement Learning Framework with Representation Learning for Concurrent Negotiation.. In *ICAART (1)*. 231–239.

[31] Yasser Mohammad. 2023. Evaluating Automated Negotiations. In *IEEE International Conference on Agents (ICA)* (Kyoto, Japan). IEEE.

[32] Yasser Mohammad. 2023. Generalized Bargaining Protocols. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 261–273.

[33] Y. Mohammad, K. Fujita, A. Greenwald, M. Klein, S. Morinaga, and S. Nakadai. 2019. *ANAC 2019 SCML*. http://tiny.cc/f8sv9y

[34] Yasser Mohammad, Amy Greenwald, and Shinji Nakadai. 2019. NegMAS: A platform for situated negotiations. In *Twelfth International Workshop on Agent-based Complex Automated Negotiations (ACAN2019) in conjunction with IJCAI* (Macau, China).

[35] Yasser Mohammad, Shinji Nakadai, and Amy Greenwald. 2019. NegMAS: A platform for situated negotiations. In *Twelfth International Workshop on Agent-based Complex Automated Negotiations (ACAN2019) in conjunction with IJCAI 2019* (Macau). https://www.github.com/yasserfarouk/negmas

[36] Yasser Mohammad, Shinji Nakadai, and Amy Greenwald. 2024. Automated Negotiation in Supply Chains A Generalist Environment for RL/MARL Research. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 19–24.

[37] Yasser Mohammad, Enrique Areyan Viqueira, Nahum Alvarez Ayerza, Amy Greenwald, Shinji Nakadai, and Satoshi Morinaga. 2019. Supply Chain Management World. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems*, Matteo Baldoni, Mehdi Dastani, Beishui Liao, Yuko Sakurai, and Rym Zalila Wenkstern (Eds.). Springer International Publishing, Cham, 153–169.

[38] Akiyuki Mori and Takayuki Ito. 2017. Atlas3: a negotiating agent based on expecting lower limit of concession function. In *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 169–173.

[39] John F Nash Jr. 1950. The bargaining problem. *Econometrica: Journal of the Econometric Society* (1950), 155–162.

[40] Yuta Ohno and Sachiyo Arai. 2024. Achieving Preferable Agreement by Utilizing offline Negotiation dialogue on Decision Transformer. In *2024 IEEE International Conference on Agents (ICA)*. IEEE, 25–30.

[41] Alexandros Papangelis and Kallirroi Georgila. 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*. 154–158.

[42] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[43] Yousef Razeghi, Celal Ozan Berk Yavuz, and Reyhan Aydoğan. 2020. Deep reinforcement learning for acceptance strategy in bilateral negotiations. *Turkish Journal of Electrical Engineering and Computer Sciences* 28, 4 (2020), 1824–1840.

[44] Bram M. Renting, Thomas M. Moerland, Holger Hoos, and Catholijn M Jonke. 2024. Towards General Negotiation Strategies with End-to-End Reinforcement Learning. *Reinforcement Learning Journal* (2024), 2059–2070.

[45] Ariel Rubinstein. 1982. Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society* 50, 1 (1982), 97–109. http://www.jstor.org/stable/1912531

[46] Ayan Sengupta, Yasser Mohammad, and Shinji Nakadai. 2021. An Autonomous Negotiating Agent Framework with Reinforcement Learning based Strategies and Adaptive Strategy Switching Mechanism. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1163–1172.

[47] Ayan Sengupta, Shinji Nakadai, and Yasser Mohammad. 2022. Transfer Learning Based Adaptive Automated Negotiating Agent Framework.. In *IJCAI*. 468–474.

[48] Toki Takahashi, Ryota Higa, Katsuhide Fujita, and Shinji Nakadai. 2022. VeNAS: Versatile Negotiating Agent Strategy via Deep Reinforcement Learning (Student Abstract). In *AAAI*, Vol. 36. 13065–13066.

[49] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 15032–15043.

[50] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032* (2024).

[51] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. 2013. *HardHeaded*. Springer Berlin Heidelberg, Berlin, Heidelberg, 223–227.

REPRODUCIBILITY

To ensure reproducibility, we provide all the code and data used in this work in the supplementary materials.

6.0.1 Code. The code is implemented in Python and provides a command-line-interface (*negmasrl*) to run the experiments reported in the paper. To run the code, you need to install it by running the following command:

> uv sync –install-extras –dev

The following command will run the first experiment in the paper:

> negmasrl scmldynamic

The following command will run the first experiment in the paper disabling reward shaping:

> negmasrl scmldynamic –no-reward-shaping

The following command will run the second experiment in the paper:

> src/scripts/run_anac.sh

The following command will run the second experiment in the paper disabling reward shaping:

> src/scripts/run_anac.sh –no-reward-shaping

To change the settings and parameters of different experiments, you can use the provided help using:

> negmasrl –help

We pinned all library versions to ensure reproducibility. The code is tested on Python 3.12 on an Appple MacBook Pro machine with an M1 chip running macOS Sequoia 15.6.1. The details of the servers used for training and testing are provided in the "Computational Resources" subsection.

This appendix describes two more experiments. The command for running each of them will be provided in the corresponding section.

If the paper is accepted, we intend to open-source the library under the name $negmas - rl$ in a public repository under GitHub with a permissive license and make it pip-installable. This will include our implementation of all comparison algorithms as well as the proposed method. The first version will match exactly the code provided in the supplementary materials. We believe that the framework will be useful in accelerating research in reinforcement learning for automated negotiation by providing a standardized way of defining problems and solutions, as well as a set of benchmarks for evaluating different approaches.

Randomization. Several of the algorithms used in this paper depend on randomization. To enhance reproducibility we seeded the random number generators in the standard Python library, numpy and torch libraries. The seed used can be found in $src/negmas - rl/cli/cli.py$ and is set to 42.

6.0.2 Data. The experiments reported in the paper – and the additional experiments reported in this appendix – use the following sets of negotiation scenarios as training and testing data:

Generalized Procurement Scenarios These are dynamically generated scenarios used in the Generalized Procurement Task (GPT) as described in the first experiment in the paper. We provide the code to generate these scenarios. For a simple example of how to generate a set of such scenarios, you can check the following script:

Table 7: Server specifications used for the experiments.

Architecture	x86_64
N. CPUs	56
Model name	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
RAM	1TB
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
Thread(s) per core	2
Core(s) per socket	14
Socket(s)	2
NUMA node(s)	2
Vendor ID	GenuineIntel
Stepping	1
CPU MHz	1240.585
BogoMIPS	5206.40
Virtualization	VT-x
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	35840K

> src/scripts/calc_opposition.py

Selected ANAC Scenarios These are the scenarios used in the second experiment in the paper (as well as one of the additional experiments in this appendix). These scenarios can all be found under the "scenarios" folder in the supplementary materials.

ANAC 2024 Scenarios These are the scenarios used in the first additional experiment in this appendix. These scenarios can all be found under the "scenarios/anac2024" folder in the supplementary materials.

EVALUATION DETAILS

Due to lack of space, some of the technical details of the evaluation are provided in this appendix.

Computational Resources

Table 7 provides the specifications of the server used for running all the experiments reported in the paper as well as the additional experiments reported in this appendix. No GPU was used in the experiments.

6.1 Evaluation Metrics

This section provides the definitions of the evaluation metrics used in the paper and this appendix. They are based the metrics proposed by Mohammad 2023a [31].

The main goal of an agent is to maximize its own **Advantage** (defined as the difference between the value it receives from negotiation and its reserved value) which can be measured for a set of Λ as: $A_a(\pi, \Lambda) \equiv \mathbb{E}_{\lambda\sim\Lambda, i\sim p_\lambda, \Upsilon\sim\mathcal{U}_i | i\approx\pi} \left[u_i(\mathcal{P}_\Upsilon(\lambda)) - u_i(\phi)\right]$.

Besides advantage, we use the following metrics to measure the quality of negotiation outcomes:

Welfare W is defined as the expected sum of the achieved value for all agents relative to the maximum achievable sum:
$W = \mathop{\mathbb{E}}_{\Lambda|\mathcal{P}(\lambda,\pi)\neq\phi} \left[\sum_{i\in\mathcal{A}[\,]} u_i(\mathcal{P}_\Upsilon(\lambda)) / \max_{\omega\in\mathbf{P}} \sum_{i\in\mathcal{A}[\,]} u_i(\omega)\right]$,

Pareto Optimality O is defined as the expected value of one minus the normalized distance of the negotiation outcomes to the rational portion of the Pareto Outcome Set for negotiations leading to agreement:
$O = \mathbb{E}_{\Lambda|\mathcal{P}(\lambda,\pi)\neq\phi} \left[1 - \min_{\omega\in\mathbf{P}\bigcup\overline{\Omega}} D(\mathcal{P}(\lambda,\pi),\omega)\right]$, where $D(a,b) \equiv \frac{\sqrt{\sum_{i\in\mathcal{A}[\,]}(u_i(a)-u_i(b))^2}}{\max_{\omega\in\Omega\wedge\psi\in\mathbf{P}\bigcup\overline{\Omega}_x}\sqrt{\sum_{i\in\mathcal{A}[\,]}(u_i(\omega)-u_i(\psi))^2}}$.

The reason we only consider the *rational* portion of the Pareto Outcome Set is to penalize negotiations ending with Pareto efficient irrational agreements.

Fairness F is defined as the expected value of one minus the normalized distance of the negotiation outcomes to any bargaining solution:
$F = \mathbb{E}_{\Lambda|\mathcal{P}(\lambda,\pi)\neq\phi} \left[1 - \min_{\omega\in\Omega^f} D(\mathcal{P}(\lambda,\pi),\phi)\right]$,
where Ω^f are the set of bargaining solutions under the single-negotiation and multiple-negotiations model (See the following section for the rationale for this definition and the details of the bargaining solutions set).

We use the following two bargaining solutions to measure fairness in this appendix:

Nash Bargaining Solution Ω_n This was the earliest of the three solutions and defined by [39] as the unique Nash Equilibrium for the Nash Bargaining Game that satisfies Perto-optimality, symmetry, scale-invariance and independence of irrelevant alternatives (IIA):

$$\Omega_n \equiv \operatorname*{argmax}_{\omega\in\mathbf{P}} \prod_{i\in\mathcal{A}} u_i(\omega) - u_i(\phi)$$

Kalai Bargaining Solution Ω_k Defined by [24] as the unique Nash Equilibrium when scale-invariance is dropped and IIA and resource monotonicity axioms are kept:

$$\Omega^*_k \equiv \operatorname*{argmax}_{\omega\in\mathbf{P}} \min_{i\in\mathcal{A}} u_i(\omega) - u_i(\phi)$$

6.2 Negotiation Scenarios

The following subsections provide more details about the negotiation scenarios used in the experiments reported in the paper and this appendix.

6.2.1 *Generalized Procurement Scenarios.*

Implemented by SCMLDynamicScenarioGenerator in src/negmas-rl/cli/scml_dynamic.py

The first experiment (Table 1 of the paper) uses the Generalized Procurement Task (GPT) scenarios. These scenarios are dynamically generated and can be used to evaluate the performance of RL methods in a wide range of negotiation problems. The scenarios are generated using the code provided in the supplementary materials (Implemented by SCMLDynamicScenarioGenerator in src/negmas-rl/cli/scml_dynamic.py). The process can be summarized as:

- Add carbon emission issue with probability 0.5
- Sample the seller utility function as follows:

Table 8: Scenario Generation Parameters for the Generalized Procurement Task. All parameters are sampled uniformly within the given range

N. Issues	(2, 3)
N. Prices	(5, 7)
N. Quantities	(10, 20)
N. Carbon Emission Settings	3
Reserved value (Seller)	(0.0, 0.2)
Reserved value (Buyer)	(0.0, 0.1)
Target Quantity	(2, Max Quantity -1)

  - Sample a target quantity uniformly within the range specified in Table 8 (less than the maximum quantity minus one).
  - The price is uniformly increasing.
  - Sample a reserved value uniformly within the range specified in Table 8.
  - If the carbon emission issue is present, sample the value of each of its three settings uniformly in the range specified in Table 8.
  - sample issue weights randomly in the range $[0.1, 1.0]$ for each issue then normalized them to sum to one.
- Sample the buyer utility function as follows:
  - Sample a target quantity uniformly within the range specified in Table 8 (less than the maximum quantity minus one).
  - The price is uniformly decreasing.
  - Sample a reserved value uniformly within the range specified in Table 8.
  - If the carbon emission issue is present, sample the value of each of its three settings uniformly in the range specified in Table 8.
  - sample issue weights randomly in the range $[0.1, 1.0]$ for each issue then normalized them to sum to one.

We then normalize both utility functions to ensure that the utility range is between zero and one.

Table 8 provides the parameters used to generate the scenarios. Fig. 4 shows some examples of the generated scenarios.

The number of outcomes will range between (50 to 420). This process generated scenarios with different levels of opposition, which is defined as the minimum distance between the Pareto-frontier and the point with maximum utility for both agents. In our case, because all utility values are normalized between zero and one, it is simply $\min_{\omega \in \Omega} \sum_{a \in \mathcal{A}} (1 - u_a(\omega))$. The reserved value is sampled uniformly within the range specified in Table 8. The target quantity is also sampled uniformly, but it is constrained to be less than the maximum quantity minus one. The number of issues, prices, and quantities are also sampled uniformly within the specified ranges. Fig. 5 shows the resulting distribution of opposition levels for the generated scenarios (based on $10,000$ samples).

6.2.2 Selected ANAC Scenarios.

. For the second experiment, we selected all the scenarios used by Sengupta et al. 2021 [46], Higa et al. 2023 [21], Takahashi et al. 2022 [48] for their evaluations of the baseline RL methods we use in the paper[3]. These scenarios were selected from the scenarios used in the ANAC competition between 2010 and 2018 to cover a wide range of outcome-space sizes and opposition levels. Table 9 provides a fuller description of these domains. We preprocessed all domains to ensure that utility values are normalized between zero and one.

6.2.3 ANAC 2024 Scenarios.

. For the first additional experiment, we used 150 scenarios out of 488 from the ANAC 2024 competition. All of these scenarios are available in the supplementary materials under the "scenarios/anac2024" folder. We selected the smallest and largest 75 scenarios based on the number of outcomes. Each was divided into 50 training scenarios and 25 testing scenarios. The number of issues ranged between 1 and 3 and the number of outcomes ranged from 729 to $1,331$. This set had an opposition level ranging from 0.208 to 0.960 with a mean of 0.627 and standard deviation of 0.122. These over a slightly larger range than the Generalized Procurement Task scenarios. Fig. 6 shows the distribution of opposition levels in this set of scenarios.

6.3 Training Parameters

We used the Soft-Actor-Critic for all experiments with the parameters given in Table 11. These are the default parameters as of the current version of stable-baselines3 (2.7.0). We did not conduct any hyper-parameter tuning and the results reported are the first results obtained on the testing set for each experiment. We used a simple MLP for all networks (the actor, critic and critic target) with two hidden layers of size 256 and ReLU activation.

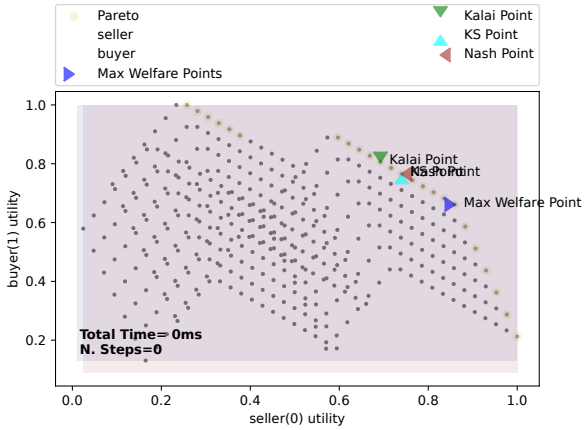
Table 10 summarizes the main characteristics of all the experiments reported in this appendix including the two reported in the paper (1, 2). Due to lack of space, the remaining experiments are only reported in this appendix. We selected experiments 1 and 2 to report in the main text because they provide examples of the hardest and easiest situations an automated negotiation RL learner faces and happen to give the best and worst performance for the proposed method. The provided code allows the user to re-run all of these experiments using:

```
negmasrl experiment-code
```

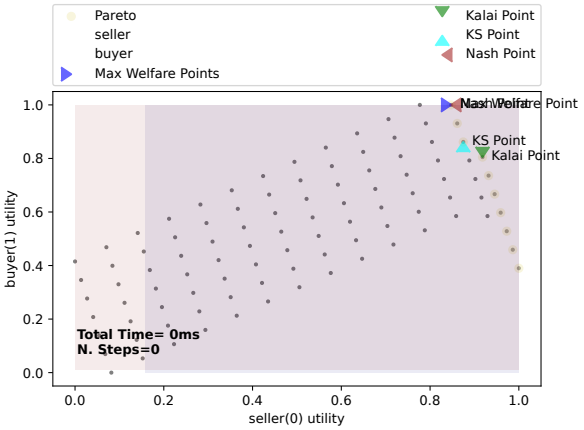
In table 10, an experiment is marked as testing Opponent Generalization (Opp. Gen.) if the agent is trained against a set of opponents and then tested against a different set of opponents. It is marked as testing Scenario Generalization (Scen. Gen.) if it is trained on a set of scenarios and then tested on a different set of scenarios given that these scenarios have the same structure (i.e., the same number of issues and the same number of values for each issue). It is marked as testing Structure Generalization (Structure Gen.) if it is trained on a set of scenarios and then tested on a different set of scenarios with a different structure (i.e., the number of issues or the number of values for each issue change all the time but have the same ranges in training and testing).

We used the same reward-shaping method for all algorithms. To encourage agreement, a penalty of -1 was added to the reservation value when calculating the reward of a negotiation ending in

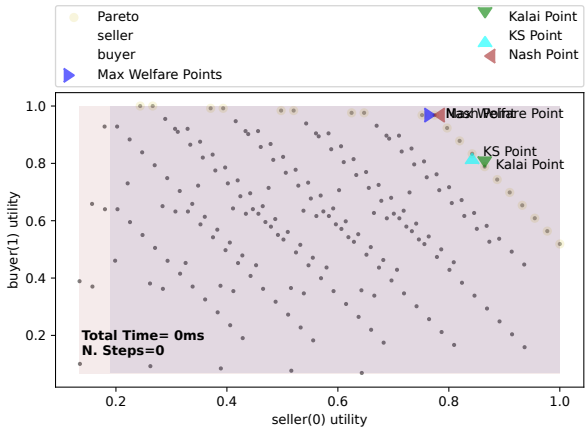
[3]The scenarios used by Bakker et al. 2019 [11] are unavailable to us.



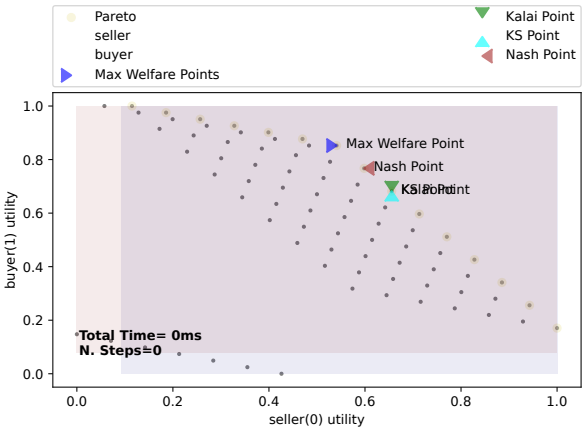
(a) Example 1



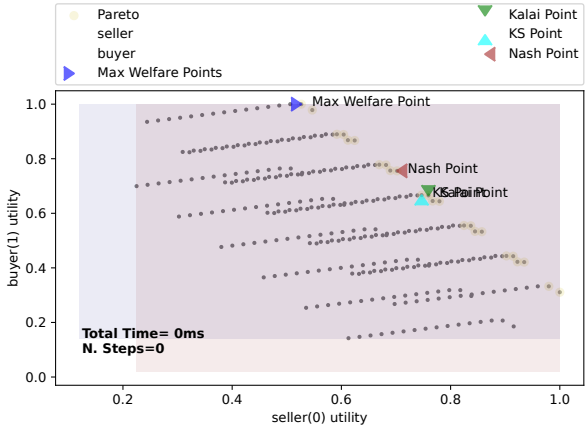
(b) Example 2



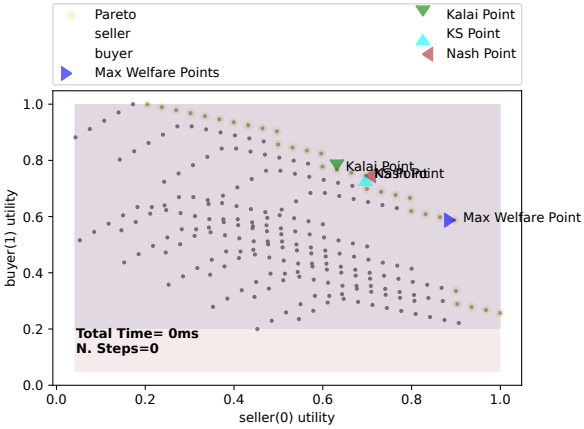
(c) Example 3



(d) Example 4



(e) Example 5



(f) Example 6

Figure 4: Example generated scenarios for the Generalized Procurement Task. Each scenario is represented by a set of issues, prices, quantities, and carbon emission settings. The reserved value and target quantity are also shown. The scenarios are dynamically generated using the parameters provided in Table 8.

Table 9: Selected ANAC Scenarios. The scenarios are selected from the ANAC competition and cover a wide range of outcome-space sizes and opposition levels. This dataset combines all the scenarios used for evaluating all of our SOTA RL method baselines.

Scenario	Used by	N. Issues	Issue Sizes	N. Outcomes	Opposition Level
Laptop	VeNAS, MiPN	3	(3, 3, 3)	27	0.276
ItexvsCypress	VeNAS, MiPN	4	(5, 4, 3, 3)	180	0.396
ISBTAcquisition	VeNAS, MiPN	5	(4, 3, 4, 4, 2)	384	0.323
Grocery	MiPN	5	(4, 5, 4, 4, 5)	1600	0.122
Thompson	MiPN	5	(5, 5, 5, 5, 5)	3125	0.276
Camera	Sengupta	6	(5, 4, 3, 3, 5, 4)	3600	0.282
Car	MiPN	6	(5, 5, 5, 5, 5, 5)	15625	0.092
EnergySmallA	MiPN	6	(5, 5, 5, 5, 5, 5)	15625	0.430

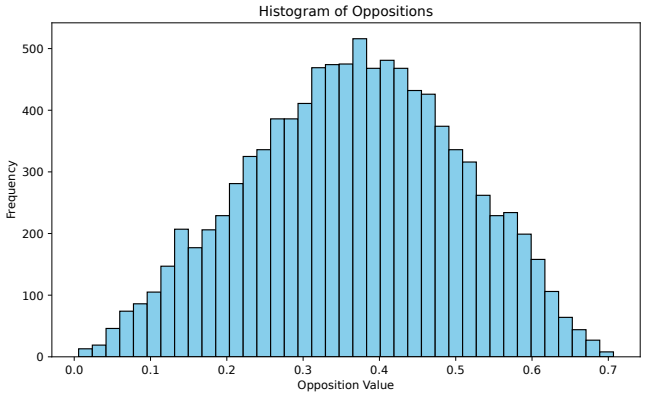


Figure 5: Opposition levels in the Generalized Procurement Task scenarios. The opposition levels are calculated as the difference between the maximum and minimum reserved values of the agents in each scenario. The histogram shows the distribution of opposition levels across all scenarios.

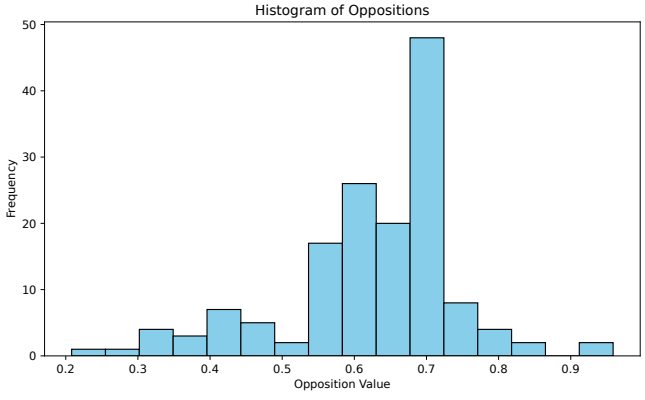


Figure 6: Opposition levels in the ANAC 2024 scenarios. The opposition levels are calculated as the difference between the maximum and minimum reserved values of the agents in each scenario. The histogram shows the distribution of opposition levels across all scenarios.

disagreement. To encourage exploration of the outcome-space, a reward of $0.002 \times t$ was given every step where t is the relative time in the negotiation (starts at zero and ends at 1). All negotiations were limited to 100 steps.

EXPERIMENTAL DETAILS

Generalized Procurement Task (GPT)

This experiment uses the Generalized Procurement Task scenarios described earlier. The goal of this experiment is to evaluate the performance of the proposed method in a dynamic setting where the opponent strategy, outcome-space structure, opposition level, and reserved values are all changing during training and testing and are different in the two cases. We ran a single training session for each method and then evaluated the performance on 50 testing scenarios. All methods were tested against the same scenarios allowing us to conduct paired hypothesis tests (i.e. Wilcoxon and t-tests) on the results. All methods were trained for $400,000$ steps with no early stopping.

We used the following state-of-the-art ANAC winner strategies as opponents for training: Atlas3 [38](winner of ANAC 2015), CUHK by [20] (winner of ANAC 2012) and two baselines (Boulware and Conceder). For testing we used the following strategies as opponents: AgentK [25],Hardheaded by [51], and AgentGG [5] (winners of ANAC 2011, 2013, and 2023), and the Linear concession time-based strategy.

Table 12 shows the model sizes of the policy for different algorithms in this experiment. The proposed method (and its variations) has a model size in between RLBOA$^+$ and Sengupta's methods. All model sizes are within 18% from each other.

Table 13 shows the value of all evaluation metrics on this experiment (the main text reports only the advantage). The proposed method outperforms all other methods in terms of advantage, welfare, Pareto optimality, and fairness (whether measured using distance to the Nash or the Kalai point). The proposed method also achieves the highest Pareto optimality and fairness scores. The results show that the proposed method is able to learn a policy that is more advantageous, fair, and Pareto optimal than the other methods under the settings of this experiment.

Table 17 shows the Wilcoxon's rank-sum test details for this experiment. The improvement in learner's advantage provided by the proposed method is statistically significant compared to all other methods with the exception of the VUDONoObs variant. The proposed

Table 10: Summary of reported experiments

#	Code	Dataset	Opp. Gen.	Scenario Gen.	Structure Gen.	Paper	Difficulty
1	scmldynamic	Generalized Procurement Task	yes	yes	yes	yes	Hard
2	anac2024	ANAC 2024 Scenarios	yes	yes	yes	yes	Hard
3	anac	Selected ANAC Scenarios	no	no	no	yes	Easy
4	mipn	Selected ANAC Scenarios	no	yes	yes	yes	Medium

Table 11: Parameters of Soft-Actor-Critic used in all experiments.

Parameter	Value
learning rate	0.0003
buffer size	1000000
learning starts	100
batch size	256
tau	0.005
gamma	0.99
train freq	1
gradient steps	1
target update interval	1
stats window size	100
Optimizer	Adam

Table 12: Model Sizes for the Generalized Procurement Task

Algorithm	N. Paramters	Algorithm	N. Paramters
RLBOA$^+$	395,526	Sengupta	341,766
Renting$^+$	467,460	VUDONoAct	349,446
VUDONoObs	346,380	VUDO	354,060

method received 0.065 higher advantage than the VUDONoObs variant, but this difference was not statistically significant under the Wilcoxon's rank-sum test. Table 19 shows the paired ttest details for this experiment which repeats the same story. Table 18 shows the Wilcoxon's rank-sum test details for the learner's utility in this experiment. The results are similar to the advantage results, with the proposed method outperforming all other methods *including* the VUDONoObs variant.

Table 20 reports the Wilcoxon's rank-sum test details for Welfare. The proposed method outperforms all other methods except VUDONoAct. Differences from SOTA heuristics were all statistically significant but difference from SOTA RL methods were not statistically significant. Table 22 reports the Wilcoxon's rank-sum test details for Pareto Optimality. The same pattern of results is observed here as well.

Considering fairness, Table 23 shows the Wilcoxon's rank-sum details for the Nash Optimality and Table 21 reports them for Kalai Optimality. Again, the proposed method outperforms all methods except VUDONoAct. The differences were statisically significant for all SOTA heuristics except RLBOA$^+$. The difference between VUDO and VUDONoObs was not statistically significant for Nash Optimality but was significant for Kalai Optimality.

6.4 ANAC 2024 Scenarios

In this experiment, we used the ANAC 2025 Scenarios described earlier. This dataset included 100 training scenarios and 50 testing scenarios. The goal of this experiment was to evaluate the performance of the proposed method in a more challenging setting compared with the previous experiments as these scenarios are less related compared with the Generalized Procurement scenarios and we conducted a single training session for each method.

We used the following state-of-the-art ANAC winner strategies as opponents for training: Atlas3 [38](winner of ANAC 2015), CUHK by [20] (winner of ANAC 2012) and two baselines (Boulware and Conceder). For testing we used the following strategies as opponents: AgentK [25],Hardheaded by [51], and AgentGG [5] (winners of ANAC 2011, 2013, and 2023), and the Linear concession time-based strategy.

We used $400,000$ steps for training and 100 steps as a round-limit for each negotiation.

Table 14 shows the results of this experiment. Here we can see that the proposed method outperformed all other methods in terms of advantage but not in terms of other metrics. This suggests that even though the agent could learn a policy that is more advantageous than the other methods for itself, the policy was not as effective in terms of welfare, Pareto optimality and fairness.

Table 24 shows the Wilcoxon's rank-sum test details for this experiment. The improvement achieved by the proposed method was statistically significant compared with its variants suggesting that the proposed codec is useful for input and output. The difference between VUDO and all other methods was statistically significant except for Sengupta and AgentK.

6.5 ANAC Scenarios, Single Opponent

The second experiment uses the selected ANAC scenarios described in Table 9. The goal of this experiment is to evaluate the performance of the proposed method against baselines in the simplest possible case with on training session per scenario and with the same opponent during training and testing (i.e., no opponent generalization). We used Boulware as the opponent because earlier studies showed that most SOTA heuristics behave mostly as Boulware agents with some small variations in their behavior added to the behavior of Boulware [46]. For each scenario we trained each model for $100,000$ steps and then tested it in 30 negotiations.

Table 13: Results of the Generalized Procurement Task experiment (1). The table shows the average advantage, welfare, Pareto optimality, and fairness for each method across all scenarios. The best results are highlighted in bold.

Method	Advantage	Welfare	Nash	Kalai	Pareto	Neg. Time(s)	Training Time(s)
Atlas3	0.154 (0.279)	0.297 (0.375)	0.425 (0.284)	0.456 (0.291)	0.505 (0.301)	0.091 (0.061)	N/A
Hardheaded	0.176 (0.354)	0.240 (0.307)	0.340 (0.179)	0.372 (0.192)	0.486 (0.277)	0.080 (0.065)	N/A
AgentK	0.201 (0.380)	0.259 (0.299)	0.349 (0.178)	0.364 (0.156)	0.494 (0.279)	0.081 (0.045)	N/A
AgentGG	0.174 (0.348)	0.238 (0.298)	0.345 (0.174)	0.378 (0.192)	0.488 (0.276)	**0.078 (0.067)**	N/A
Sengupta	0.191 (0.328)	0.323 (0.379)	0.448 (0.292)	0.461 (0.278)	0.521 (0.298)	0.153 (0.092)	**8,927**
RLBOA$^+$	0.160 (0.306)	0.279 (0.349)	0.389 (0.251)	0.423 (0.261)	0.495 (0.283)	0.184 (0.068)	11,105
Renting$^+$	0.230 (0.253)	0.378 (0.277)	0.475 (0.214)	0.500 (0.200)	0.571 (0.208)	0.71 (0.19)	48,158
VUDO	**0.298 (0.272)**	**0.683 (0.329)**	**0.678 (0.239)**	**0.641 (0.204)**	**0.803 (0.218)**	0.112 (0.075)	9,063
VUDONoAct	0.215 (0.319)	0.376 (0.407)	0.497 (0.325)	0.515 (0.323)	0.560 (0.313)	0.146 (0.082)	9,008
VUDONoObs	0.223 (0.342)	0.417 (0.393)	0.467 (0.254)	0.491 (0.267)	0.615 (0.305)	0.127 (0.076)	8,968

Table 14: Results of the selected ANAC 2024 scenarios experiment (2). The table shows the average advantage, welfare, Pareto optimality, and fairness for each method across all scenarios. The best results are highlighted in bold.

Method	Advantage	Welfare	Nash	Kalai	Pareto	Neg. Time
Atlas3	0.163 (0.300)	0.222 (0.398)	0.522 (0.209)	0.560 (0.173)	0.618 (0.209)	0.538 (0.204)
AgentK	0.153 (0.350)	0.158 (0.362)	0.454 (0.140)	0.490 (0.094)	0.580 (0.188)	0.215 (0.067)
AgentGG	0.040 (0.196)	0.040 (0.196)	0.448 (0.133)	0.485 (0.088)	0.523 (0.110)	0.153 (0.056)
Hardheaded	0.040 (0.196)	0.040 (0.196)	0.448 (0.133)	0.485 (0.088)	0.523 (0.110)	0.129 (0.055)
RLBOA$^+$	0.077 (0.141)	0.212 (0.367)	0.504 (0.186)	0.546 (0.150)	0.614 (0.197)	0.228 (0.064)
Sengupta	0.117 (0.251)	0.887 (0.123)	0.529 (0.149)	0.544 (0.095)	0.992 (0.030)	0.179 (0.059)
Renting$^+$	0.018 (0.120)	0.451 (0.246)	**0.635 (0.204)**	**0.653 (0.206)**	0.694 (0.182)	3.952 (1.908)
VUDO	**0.179 (0.323)**	0.227 (0.406)	0.511 (0.196)	0.549 (0.156)	0.622 (0.216)	0.160 (0.054)
VUDONoAct	0.048 (0.111)	0.334 (0.429)	0.495 (0.155)	0.530 (0.115)	0.687 (0.234)	0.175 (0.078)
VUDONoObs	0.050 (0.016)	**0.891 (0.123)**	0.481 (0.113)	0.495 (0.024)	**0.998 (0.005)**	**0.070 (0.027)**

Table 15: Results of the selected ANAC scenarios experiment (3). The table shows the average advantage, welfare, Pareto optimality, and fairness for each method across all scenarios. The best results are highlighted in bold.

Method	Advantage	Welfare	Nash	Kalai	Pareto	Neg. Time (s)	Training Time (s)
AgentGG	0.800 (0.422)	0.699 (0.377)	0.605 (0.255)	0.574 (0.227)	0.864 (0.286)	0.304 (0.359)	N/A
AgentK	**0.947 (0.049)**	**0.910 (0.080)**	**0.795 (0.141)**	0.763 (0.122)	**0.990 (0.019)**	0.388 (0.296)	N/A
Atlas3	0.684 (0.130)	0.868 (0.091)	0.855 (0.077)	**0.872 (0.073)**	0.930 (0.063)	0.364 (0.396)	N/A
Hardheaded	0.800 (0.422)	0.699 (0.377)	0.605 (0.255)	0.574 (0.227)	0.864 (0.286)	**0.293 (0.333)**	N/A
VeNAS(c)	0.541 (0.372)	0.706 (0.457)	0.694 (0.327)	0.707 (0.320)	0.803 (0.329)	0.340 (0.367)	8831
MiPN(c)	0.475 (0.132)	0.860 (0.070)	0.733 (0.080)	0.769 (0.060)	0.987 (0.022)	0.297 (0.346)	N/A
RLBOA$^+$	0.846 (0.346)	0.782 (0.333)	0.702 (0.234)	0.675 (0.215)	0.919 (0.229)	0.415 (0.300)	3755
Sengupta	0.828 (0.304)	0.831 (0.255)	0.723 (0.216)	0.699 (0.201)	0.959 (0.163)	0.323 (0.260)	4,006
Renting$^+$	0.636 (0.385)	0.588 (0.356)	0.539 (0.231)	0.548 (0.232)	0.784 (0.286)	0.486 (0.470)	5,214
VUDO	0.915 (0.175)	0.867 (0.107)	0.727 (0.154)	0.701 (0.140)	0.991 (0.024)	0.517 (0.487)	3,173
VUDO-NoAct	0.856 (0.209)	0.876 (0.149)	0.751 (0.197)	0.751 (0.201)	0.989 (0.072)	0.326 (0.274)	**2,863**
VUDO-NoObs	0.824 (0.340)	0.863 (0.108)	0.707 (0.164)	0.683 (0.154)	1.000 (0.003)	0.442 (0.334)	4,133

Table 16: Results of the Selected ANAC Scenarios with scenario generalization experiment (4). The table shows the average advantage, welfare, Pareto optimality, and fairness for each method across all scenarios. The best results are highlighted in bold.

Scenario	AgentGG	AgentK	Atlas3	Hardheaded	RLBOA$^+$	Sengupta	Renting$^+$	VUDO
Advantage	0.720 (0.449)	**0.959 (0.042)**	0.668 (0.115)	0.720 (0.449)	0.776 (0.347)	0.726 (0.265)	0.478 (0.255)	0.893 (0.163)
Welfare	0.611 (0.390)	**0.892 (0.074)**	0.870 (0.100)	0.611 (0.390)	0.756 (0.322)	0.853 (0.088)	0.486 (0.348)	0.878 (0.101)
Nash	0.528 (0.246)	0.746 (0.123)	**0.866 (0.082)**	0.528 (0.246)	0.669 (0.210)	0.673 (0.143)	0.502 (0.192)	0.735 (0.142)
Kalai	0.514 (0.229)	0.733 (0.113)	**0.875 (0.080)**	0.514 (0.229)	0.656 (0.201)	0.691 (0.136)	0.506 (0.193)	0.717 (0.134)
Pareto	0.810 (0.305)	0.994 (0.011)	0.928 (0.070)	0.810 (0.305)	0.909 (0.225)	**0.999 (0.007)**	0.696 (0.288)	0.991 (0.025)
Neg. Time	0.291 (0.342)	0.332 (0.309)	0.328 (0.372)	0.278 (0.331)	0.401 (0.326)	**0.276 (0.297)**	1.520 (1.153)	0.449 (0.451)
Training Time	N/A	N/A	N/A	N/A	5,521.618	**4,440.505**	5,979.667	4,627.216

Table 17: Wicoxon's rank-sum test details for learner advantage on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	p-value	statistic	significant
Atlas3	0.002158	929	True
AgentK	0.034651	826	True
AgentGG	0.017255	734	True
Hardheaded	0.015903	737	True
RLBOA$^+$	0.003778	785	True
Sengupta	0.021376	697	True
Renting$^+$	0.012011	702	True
VUDONoObs	0.090463	747	False
VUDONoAct	0.032481	681	True

Table 18: Wicoxon's rank-sum test details for learner utility on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	pvalue	statistic	significant
Atlas3	0.001418	2.985024	True
AgentK	0.002974	2.750634	True
AgentGG	0.000424	3.336609	True
Hardheaded	0.000384	3.364184	True
RLBOA$^+$	0.000542	3.267671	True
Sengupta	0.006227	2.499010	True
Renting$^+$	0.000061	3.614459	True
VUDONoObs	0.023872	1.713035	True
VUDONoAct	0.024684	1.685460	True

Table 15 shows the results of this experiment. Here we can see that the proposed method (and ALL RL based methods) did not perform as well as the SOTA heuristic (AgnetK). Nevertheless, Utility-Projection methods outperformed all other SOTA heuristics and the proposed method outperformed all all other SOTA heuristics, all RL based SOTA methods as well as the variations of the proposed method.

Table 25 shows the Wilcoxon's rank-sum test details for this experiment comparing the proposed method with all other methods.

Table 19: Paired t-test test details for learner advantage on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

b	pvalue	statistic	significant
Atlas3	0.001956	3.028588	True
AgentK	0.048410	1.692927	True
AgentGG	0.018506	2.144105	True
Hardheaded	0.017357	2.172133	True
RLBOA$^+$	0.005659	2.632045	True
Sengupta	0.021075	2.086601	True
Renting$^+$	0.010123	2.244853	True
VUDONoObs	0.102410	1.285039	False
VUDONoAct	0.042268	1.760663	True

Table 20: Wilcoxon's rank-sum test details for learner Welfare on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	p-value	statistic	Significant
Atlas3	0.000113	3.688194	True
AgentK	0.000006	4.384469	True
AgentGG	0.000001	4.836015	True
Hardheaded	0.000001	4.801546	True
Sengupta	0.071469	1.464937	False
RLBOA$^+$	0.056810	1.582132	False
VUDONoAct	0.832761	-0.965135	False
VUDONoObs	0.085052	1.371870	False

Even though AgentK outperforms the proposed method with an advantage of 0.947 compared to 0.915 for VUDO, the difference is not statistically significant. The proposed method outperforms all other methods including its two variants. Nevertheless, the differences between the proposed method and Utility Projection methods and its variants were not statistically significant. Note that, for this experiment, the number of samples equals the number of scenarios which is 8. This means that the Wilcoxon's rank-sum test results should be taken with caution as the number of samples is too small. Using multiple runs for each scenario would not have been better

Table 21: Wilcoxon's rank-sum test details for learner Kalai Optimality (fairness) on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	pvalue	statistic	significant
Atlas3	0.001266	3.019493	True
AgentK	0.000001	4.887718	True
AgentGG	0.000002	4.608519	True
Hardheaded	0.000002	4.642988	True
RLBOA$^+$	0.021167	2.030230	True
Sengupta	0.109246	1.230547	False
VUDONoAct	0.801764	-0.847940	False
VUDONoObs	0.096889	1.299485	True

Table 22: Wilcoxon's rank-sum test details for learner Pareto Optimality on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	p-value	statistic	Significant
Atlas3	0.000291	3.440016	True
AgentK	0.000041	3.936371	True
AgentGG	0.000075	3.791601	True
Hardheaded	0.000069	3.812282	True
RLBOA(c)	0.049715	1.647623	False
Sengupta	0.089985	1.340848	False
VUDONoAct	0.846203	-1.020285	False
VUDONoObs	0.142638	1.068542	False

Table 23: Wilcoxon's rank-sum test details for learner Nash Optimality (fairness) on the Generalized Procurement Task experiment (1) comparing VUDO with all other methods

Against	p-value	statistic	Significant
Atlas3	0.000169	3.584786	True
AgentK	0.000000	4.915294	True
AgentGG	0.000000	4.925634	True
Hardheaded	0.000000	4.966997	True
RLBOA$^+$	0.019639	2.061252	True
Sengupta	0.125521	1.147821	False
VUDONoAct	0.821299	-0.920325	False
VUDONoObs	0.081357	1.395999	True

because most strategies behave deterministically and repeated runs tend to lead to the same results.

6.6 Selected ANAC Scenarios, Generalization

In this experiment, we use the selected ANAC scenarios used in the second experiment. This time, we train using 5 scenarios and test on the remaining 3. We used the same training and testing opponents as in the second experiment. This experiment is of medium difficulty as the agent needs to learn to generalize over the scenarios but not over the opponent strategies.

Table 16 shows the results of this experiment comparing the proposed method with SOTA heuristics and SOTA RL methods. The proposed method outperforms all comparison methods in terms of advantage except Agent (which is expected given the results of the second experiment).

Table 26 presents the ablation study for this scenario set. The proposed method outperformed both variations in all metrics except

Table 24: Wicoxon's rank-sum test details for learner advantage on the ANAC 2024 scenarios (2) without opponent or scenario generalization comparing VUDO with all other methods

Against	pvalue	statistic	significant
Atlas3	0.020430	56	True
AgentK	0.406973	42	False
AgentGG	0.002371	75	True
Hardheaded	0.002371	75	True
Sengupta	0.614824	204	False
RLBOA$^+$	0.001109	78	True
VUDONoObs	0.004998	174	True
VUDONoAct	0.001081	114	True

Table 25: Wicoxon's rank-sum test details for learner advantage on the Selected ANAC scenarios (3) without opponent or scenario generalization comparing VUDO with all other methods

Against	p-value	statistic	significant
Atlas3	0.030119	476	True
AgentK	1.000000	10	False
AgentGG	0.999996	73	False
Hardheaded	0.999996	73	False
RLBOA$^+$	1.000000	44	False
Sengupta	0.999999	45	False
VeNAS(c)	0.000357	545	True
MiPN(c)	0.000000	703	True
VUDONoObs	1.000000	0	False
VUDONoAct	0.999999	40	False

Table 26: Ablation study on Selected ANAC Scenarios with scenario generalization experiment (4).

Scenario	VUDO	VUDONoAct	VUDONoObs
Advantage	**0.893 (0.163)**	0.841 (0.260)	0.780 (0.324)
Welfare	**0.878 (0.101)**	0.841 (0.199)	0.782 (0.308)
Nash	**0.735 (0.142)**	0.709 (0.182)	0.698 (0.218)
Kalai	**0.717 (0.134)**	0.696 (0.175)	0.684 (0.206)
Pareto	**0.991 (0.025)**	0.971 (0.127)	0.921 (0.210)
Neg. Time	0.449 (0.451)	**0.293 (0.288)**	0.452 (0.479)
Training Time	4627.216	**4411.329**	4759.614

negotiation time suggesting that the proposed codec method is useful for observation and action spaces.

Because we have only 3 testing scenarios, we could not conduct hypothesis tests for this experiment.