# A Contextual Combinatorial Bandit Approach to Negotiation

Yexin Li [1] Zhancun Mu [2] Siyuan Qi [1]

## Abstract

Learning effective negotiation strategies poses two key challenges: the exploration-exploitation dilemma and dealing with large action spaces. However, there is an absence of learning-based approaches that effectively address these challenges in negotiation. This paper introduces a comprehensive formulation to tackle various negotiation problems. Our approach leverages contextual combinatorial multi-armed bandits, with the *bandits* resolving the exploration-exploitation dilemma, and the *combinatorial* nature handles large action spaces. Building upon this formulation, we introduce NegUCB, a novel method that also handles common issues such as partial observations and complex reward functions in negotiation. NegUCB is contextual and tailored for full-bandit feedback without constraints on the reward functions. Under mild assumptions, it ensures a sub-linear regret upper bound. Experiments conducted on three negotiation tasks demonstrate the superiority of our approach.

## 1. Introduction

Negotiation serves as a fundamental process that underpins interaction among diverse agents across a wide spectrum of domains, ranging from diplomacy (Paquette et al., 2019; FAIR et al., 2022) and resource allocation (Lewis et al., 2017; Cao et al., 2018) to trading (Bagga et al., 2020). In these scenarios, an agent, represented as negotiator $a$, engages in negotiation with various counterparts $g$, with its state evolving. At each time step, negotiator $a$ proposes a bid and receives feedback indicating whether the counterpart $g$ accepts or rejects the proposal. Successful acceptance leads to a deal, while rejection leads to termination or further negotiation, possibly with counter-proposals from the counterpart. These negotiations can vary in form, and Figure 1 illustrates three representative negotiation problems: trading, resource allocation, and multi-issue negotiation. As negotiation experiences accumulate, an agent should continuously improve its negotiation ability.

However, effectively exploiting past experiences in subsequent negotiations is challenging in the following aspects. **Exploration-exploitation dilemma**: As counterparts vary and the agent's state evolves, over-exploiting historical data may result in sub-optimal performance, while excessive exploration may make the counterpart lose patience. Existing works on negotiation (Lewis et al., 2017; Liu & Zheng, 2020; Sengupta et al., 2022) tend to neglect exploration, primarily focusing on exploitation, or simply explore by UCT (Buron et al., 2019), without considering observable contexts. **Large action spaces**: Consider a trading task in which our negotiator possesses items $V_1$ while the counterpart holds items $V_2$. The potential bid can be any subset of the union $V = V_1 \cup V_2$, resulting in $2^{|V|}$ possible choices. Some studies (Cao et al., 2018; Bakker et al., 2019; Bagga et al., 2020) employ reinforcement learning to acquire negotiation strategies, but they primarily focus on tasks involving action spaces limited to a few hundred discrete actions or low-dimensional continuous action spaces. **Partial observations**: The profiles of counterparts, including their preferences and desires, cannot be fully observed. Relying solely on observable contexts for negotiation can be ineffective. **Complicated acceptance functions**: Inferring the likelihood of the counterpart accepting a bid remains challenging, even when their hidden states are known.

In this paper, we formulate negotiation problems using contextual combinatorial multi-armed bandits (Li et al., 2010; Chen et al., 2013; Qin et al., 2014; Wen et al., 2015; Chen et al., 2018; Agarwal et al., 2021; Nie et al., 2022) to address the exploration-exploitation dilemma and handle the large action spaces of combinatorial cardinality. Although negotiation involves a series of actions, unlike in reinforcement learning, where actions may lead to state transitions, bid actions in negotiation do not inherently trigger such transitions. Agents accumulate knowledge about their counterparts through interactions. Consequently, the bandit-based formulation is well-suited for negotiation problems.

In our formulation, an **arm** denotes an item involved in the

[1] State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China [2] Peking University. Correspondence to: Siyuan Qi <syqi@bigai.ai>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

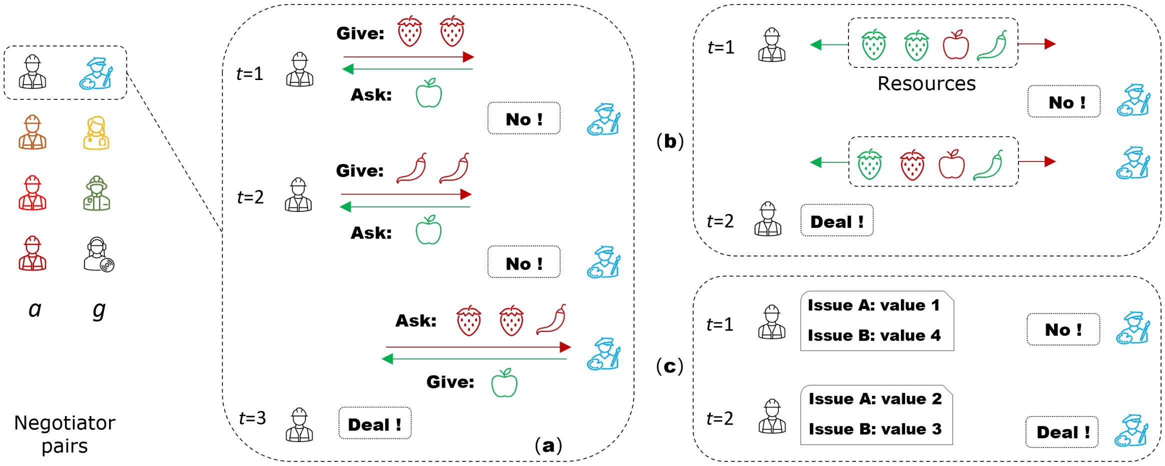


*Figure 1.* Three typical types of negotiation. Negotiator $a$ is represented with the same icon but in varying colors, indicating the same agent whose state evolves. Negotiator $g$ is depicted with distinct icons and colors, meaning different counterparts. (a) illustrates a trading task, where items in **Red** signify those that negotiator $a$ gives to $g$, while items in **Green** indicate those that counterpart $g$ gives to $a$. (b) presents a resource allocation task. Items in green are proposed for allocation to negotiator $a$, while those in red are suggested for assignment to negotiator $g$. Lastly, (c) portrays a multi-issue negotiation task involving two distinct issues, each offering several value choices. Negotiators $a$ and $g$ aim to agree on the values of these two issues.

negotiation, while a **super arm** signifies a bid composed of multiple items. The term **acceptance** is specifically designated to represent the reward, with a value of 1 assigned when the counterpart accepts the bid and 0 assigned in the case of rejection. Consequently, our primary objective is the systematic selection of super arms to gain a comprehensive understanding of the expected acceptance of each super arm while ensuring a substantial cumulative benefit in the long run. This formulation involves *full-bandit* feedback, where information regarding the acceptance of individual items within the bid remains inaccessible, and only an aggregate acceptance value for the entire bid is available. Otherwise, the feedback is referred to as *semi-bandit*. Presently, most works (Qin et al., 2014; Wen et al., 2015; Chen et al., 2018; Hwang et al., 2023) on combinatorial bandits rely on semi-bandit feedback. Although there are works (Rejwan & Mansour, 2020; Agarwal et al., 2021; Nie et al., 2022; Fourati et al., 2023) that consider full-bandit feedback, they are non-contextual and often subject to specific constraints.

Building upon the above formulation, we propose a contextual algorithm for full-bandit feedback, named **Neg**otiation **UCB** (NegUCB), to learn negotiation strategies and adeptly address the exploitation-exploration dilemma and the challenge of large action spaces. Moreover, NegUCB incorporates hidden states to tackle the issue of partial observations and handles diverse acceptance functions through kernel regression (Schulz et al., 2018; Vakili et al., 2023). Under mild assumptions, NegUCB's regret upper bound is guaranteed to be sub-linear with respect to the number of negotiation steps and independent of the bid cardinality, distinguishing itself from existing works on either semi-bandit or full-bandit feedback.

In summary, this paper makes three major contributions. First, we provide a comprehensive formulation for diverse types of negotiation problems in § 3.1. Second, we propose NegUCB to learn negotiation strategies, effectively addressing the prevalent challenges in negotiation in § 3.2. Lastly, we provide theoretical insights in § 3.3 and conduct experiments on representative negotiation tasks in § 4, highlighting the advantages and effectiveness of our method.

## 2. Related Work

### 2.1. Negotiation

Deep reinforcement learning has been applied to learning negotiation strategies. For instance, Rodriguez-Fernandez et al. (Rodriguez-Fernandez et al., 2019) adopt a DQN-based model (Mnih et al., 2015) to solve the contract negotiation problem characterized by discrete state and action spaces. Lewis et al. (Lewis et al., 2017) combine supervised learning with reinforcement learning to acquire negotiation strategies in a resource allocation task. RLBOA (Bakker et al., 2019) discretizes continuous action and state spaces and employs tabular Q-learning to learn bidding strategies, although it may encounter issues related to the curse of dimensionality. ANEGMA (Bagga et al., 2020) uses actor-critic (Bhatnagar et al., 2009) to mitigate the dimensionality challenge. Cao et al. (Cao et al., 2018) design two communication protocols to explore the emergence of communication when two agents negotiate. However, these approaches struggle to handle large discrete action spaces (Dulac-Arnold et al., 2016) and often give minimal consideration to exploration.

Some studies investigate alternative approaches to negotiation. For instance, Buron et al. learn bidding strategies relying on Monte Carlo tree search (Buron et al., 2019). A decision tree-based negotiation assistant (Liu & Zheng, 2020) is specifically designed to predict prices in a car trading platform. Sengupta et al. (Sengupta et al., 2022) demonstrate a transfer learning-based solution to adapt base negotiation strategies to new counterparts rapidly. Cicero (FAIR et al., 2022) achieves mastery in the game of *Diplomacy* by integrating reinforcement learning with a language model. Nevertheless, these approaches deal with highly specific problems or issues in negotiation, yet they have not effectively tackled the prevalent challenges discussed above.

### 2.2. Multi-Armed Bandits

LinUCB (Li et al., 2010) has been introduced to formulate *recommendation* as a contextual bandit problem, assuming linearity in the reward concerning user and item contexts. It has demonstrated effective performance in *recommendation* and guarantees a sub-linear regret bound (Chu et al., 2011). FactorUCB (Wang et al., 2017) also makes a linearity assumption but considers hidden features alongside the observable contexts, leading to an improved click rate in *recommendation*. To overcome the linearity assumption in contextual bandits, KernelUCB (Valko et al., 2013; Chowdhury & Gopalan, 2017) transforms contexts into a high-dimensional space and applies LinUCB in this new space. Neural-UCB (Zhou et al., 2020) attempts to leverage deep neural networks to capture the relationship between contexts and rewards. However, its computational complexity makes it challenging to generalize to real tasks.

CUCB (Chen et al., 2013) establishes a general framework for combinatorial multi-armed bandits. C2UCB (Qin et al., 2014) and ComLinUCB (Wen et al., 2015) incorporate contexts into combinatorial bandits based on the same linearity assumption as LinUCB. CC-MAB (Chen et al., 2018) focuses on problems with volatile arms and submodular reward functions. CN-UCB (Hwang et al., 2023) employs neural networks to address contextual combinatorial bandit problems, facing the computational limitation as Neural-UCB. However, these algorithms operate within semi-bandit feedback. Another relevant setting is the full-bandit feedback, in which rewards for individual arms are inaccessible. Algorithms designed for full-bandit feedback include CSAR (Rejwan & Mansour, 2020), DART (Agarwal et al., 2021), ETCG (Nie et al., 2022), and RGL (Fourati et al., 2023). However, their reward functions adhere to linearity or sub-modularity, and none of them consider contexts. In contrast, NegUCB is contextual, combinatorial, and tailored for full-bandit feedback without constraints on the reward functions. A comparative analysis is presented in Table 1.

*Table 1.* Comparison between NegUCB and representative multi-armed bandit algorithms. **Contextual**: consider contexts. **Combinatorial**: consider super arms consisting of multiple basis arms. **Partial**: contexts are partially observable. **Non-linear**: non-linear reward functions w.r.t. contexts. **Full-bandit**: rewards of basis arms are not available. *Blank* means the attribute does not apply.

| Algorithm | Contextual | Combinatorial | Partial | Non-Linear | Full-bandit |
|---|---|---|---|---|---|
| LinUCB | ✓ | ✗ | ✗ | ✗ | |
| FactorUCB | ✓ | ✗ | ✓ | ✗ | |
| KernelUCB | ✓ | ✗ | ✗ | ✓ | |
| Neural-UCB | ✓ | ✗ | ✗ | ✓ | |
| CUCB | ✗ | ✓ | | | ✗ |
| C2UCB | ✓ | ✓ | ✗ | ✗ | ✗ |
| ComLinUCB | ✓ | ✓ | ✗ | ✗ | ✗ |
| CC-MAB | ✓ | ✓ | ✗ | ✓ | ✗ |
| CN-UCB | ✓ | ✓ | ✗ | ✓ | ✗ |
| CSAR | ✗ | ✓ | | | ✓ |
| DART | ✗ | ✓ | | | ✓ |
| ETCG | ✗ | ✓ | | | ✓ |
| RGL | ✗ | ✓ | | | ✓ |
| **NegUCB** | ✓ | ✓ | ✓ | ✓ | ✓ |

## 3. Methodology

Unless otherwise specified, uppercase symbols represent sets, bold uppercase symbols denote matrices, bold lowercase symbols represent vectors, and lowercase symbols denote scalars or functions. $\boldsymbol{I}_d$ refers to an identity matrix with dimensions $d \times d$, and $\mathbf{0}_d$ represents a zero vector of size $d \times 1$. Kronecker product is denoted as $\otimes$. Frobenius norm of a matrix and the $l_2$ norm of a vector are respectively denoted as $\|\boldsymbol{X}\|$ and $\|\boldsymbol{x}\|$. Mahalanobis norm of a column vector $\boldsymbol{x}$ based on matrix $\boldsymbol{A}$ is denoted as $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}}$. $\text{vec}(\boldsymbol{A})$ is the vectorization operator of matrix $\boldsymbol{A}$.

### 3.1. Negotiation Formulation

In this section, we provide a comprehensive formulation that applies to various types of negotiation problems. First, we outline the negotiation framework, detailing the strategy for making proposals when it is our turn to bid and the criteria for deciding whether to accept or reject a bid from the counterpart. Next, we formulate the critical component within this framework.

#### 3.1.1. Negotiation Framework

Denote the pool of the counterpart negotiators as $U$, with a cardinality of $|U| = m$. The item pool is represented as $V$, where $|V| = n$. It is essential to acknowledge that negotiation with a new counterpart may occur at any time, leading to an increase in $m$ over time. Additionally, new items may be added to $V$. Without loss of generality, we assume these two pools $U$ and $V$ to be constant. At time step $\tau$, our negotiator has a valid bid set $B_\tau$ encompassing all feasible bids it can propose at this time. For example, in a trading task, a bid specifies which items our negotia-

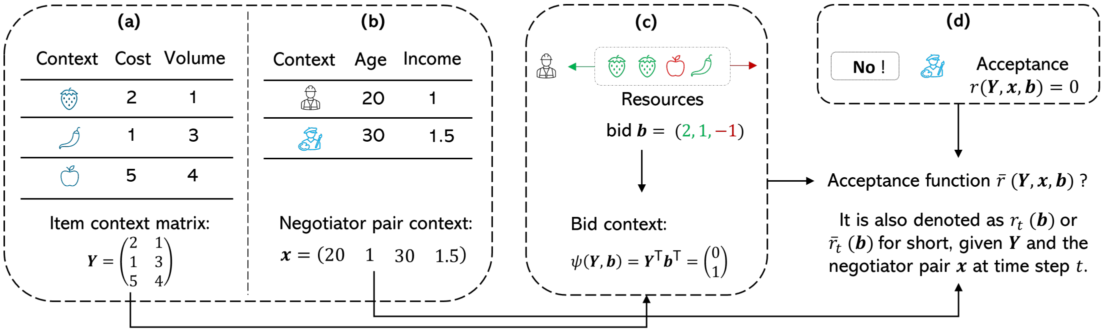

(a)
Context Cost Volume
2 1
1 3
5 4
Item context matrix:
$Y = \begin{pmatrix} 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix}$
(b)
Context Age Income
20 1
30 1.5
Negotiator pair context:
$x = (20 \quad 1 \quad 30 \quad 1.5)$
(c)
Resources
bid $b = (2, 1, -1)$
Bid context:
$\psi(Y, b) = Y^{\mathrm{T}} b^{\mathrm{T}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
(d)
No !
Acceptance
$r(Y, x, b) = 0$
Acceptance function $\bar{r}\,(Y, x, b)$ ?
It is also denoted as $r_t\,(b)$ or $\bar{r}_t\,(b)$ for short, given $Y$ and the negotiator pair $x$ at time step $t$.

*Figure 2.* Acceptance function of a resource allocation task. (a) and (b) describe the contexts of the items and the current negotiator pair, respectively. (c) provides an example illustrating how the bid can be defined and how to extract the bid context. (d) depicts the acceptance label. The goal is to approximate the acceptance function $\bar{r} : (\boldsymbol{Y}, \boldsymbol{x}, \boldsymbol{b}) \mapsto r$ using historical negotiation data.

tor proposes to give to the counterpart and which items it requests in return. The validity of a bid is determined by whether our negotiator possesses the items it proposes to give. More designation of bids under various negotiation scenarios will be elaborated later.

For a valid bid $\boldsymbol{b} \in B_\tau$ of our negotiator at time $\tau$, the acceptance function $r_\tau$ assesses whether the counterpart may accept or reject the bid, denoted by $r_\tau(\boldsymbol{b}) = 1$ or $r_\tau(\boldsymbol{b}) = 0$. This function is unknown and needs to be learned. Additionally, there exists a benefit function $f_\tau$ that measures the potential benefit of the bid to our negotiator, a metric highly dependent on the specific problem. Consequently, the optimal bid for our negotiator to propose at time step $\tau$ is determined by Equation 1, aiming to maximize its expected benefit. During negotiation, if our negotiator intends to propose a bid, it chooses a valid bid using Equation 1. Supposing our negotiator receives a bid $\boldsymbol{b}$ from the counterpart, it is evident that $r_\tau(\boldsymbol{b}) = 1$; thus, our negotiator decides whether to accept the bid by evaluating if the bid is valid and optimal after setting $r_\tau(\boldsymbol{b}) = 1$.

$$\boldsymbol{b}_\tau^* = \arg\max r_\tau(\boldsymbol{b}) \times f_\tau(\boldsymbol{b}) \tag{1}$$

### 3.1.2. Acceptance Function

As the benefit function $f_\tau$ is dependent on the specific problem and crafted manually, it is not the primary focus of this work. Instead, we focus on learning the acceptance function $r_\tau$ using contextual combinatorial multi-armed bandits (Chen et al., 2013; Qin et al., 2014; Wen et al., 2015; Chen et al., 2018; Nie et al., 2022), where **arms** represent items in $V$, **super arms** denote bids, and rewards are the **acceptance** labels. Consequently, the objective is to iteratively put forth beneficial bids to understand the expected acceptance of each bid by various counterparts while ensuring a substantial cumulative benefit in the long run. In the following, we review the details based on Figure 2.

Items in pool $V$ have contexts denoted as row vectors $\{\boldsymbol{y}_w | w = 1, 2, ..., n\}$, collectively forming an item context matrix $\boldsymbol{Y} = [\boldsymbol{y}_1; \boldsymbol{y}_2; ...; \boldsymbol{y}_n]$, as depicted in Figure 2 (a). At time step $\tau$, our negotiator and the counterpart form a negotiator pair, characterized by contexts denoted as a row vector $\boldsymbol{x}_\tau$, as depicted in Figure 2 (b). It is worth noting that $\boldsymbol{x}_\tau$ corresponds to one of the $m$ counterparts in $U$. In other words, it is a row of the negotiator context matrix $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; ...; \boldsymbol{x}_m]$. In this work, we use $\boldsymbol{x}_\tau$ or $\boldsymbol{x}_w, w = 1, 2, ..., m$, interchangeably to either highlight the time step or the counterpart index. Addressing the partial observation issue, we assume hidden states $\boldsymbol{U} = [\boldsymbol{u}_1; \boldsymbol{u}_2; ...; \boldsymbol{u}_m]$ for the $m$ negotiator pairs. Then the acceptance function $r_\tau$ at time step $\tau$ is estimated through Equation 2, where $\boldsymbol{\Theta}$ in the first term represents the function parameters, $\boldsymbol{u}_\tau$ in the second term signifies the hidden state of the current negotiator pair. Specifically, the first term estimates the partial acceptance of the bid based on observed contexts, while the second term evaluates the partial acceptance of the bid based on hidden states.

$$\bar{r}_\tau(\boldsymbol{b}_\tau) = \phi(\boldsymbol{x}_\tau)\boldsymbol{\Theta} \left\langle \boldsymbol{Y}, \boldsymbol{b}_\tau \right\rangle + \boldsymbol{u}_\tau \left\langle \boldsymbol{Y}, \boldsymbol{b}_\tau \right\rangle \tag{2}$$

$$\left\langle \boldsymbol{Y}, \boldsymbol{b}_\tau \right\rangle = \phi \circ \psi(\boldsymbol{Y}, \boldsymbol{b}_\tau) \tag{3}$$

In the first term of Equation 2, function $\phi$ transforms context $\boldsymbol{x}_\tau$ into a $h$-dimensional space $\mathcal{H}$ where $h$ can be infinite. $\langle \boldsymbol{Y}, \boldsymbol{b}_\tau \rangle$ is expressed in Equation 3, where function $\psi$ extracts the context of bid $\boldsymbol{b}_\tau$ from the item context matrix $\boldsymbol{Y}$. A possible example of $\psi$ is provided in Figure 2 (c). Following this, function $\phi$ further transforms the bid context into a high-dimensional representation within space $\mathcal{H}$. Specifically, function $\phi$ transforms contexts into high-dimensional representations, allowing the acceptance function to operate non-linearly concerning the observed contexts.

Given historical negotiation data from step $1, 2, ..., \tau$, we aim to optimize the following objective function to derive functions $\psi$ and $\phi$, parameters $\boldsymbol{\Theta}$ and hidden states $\boldsymbol{U}$, then use them for the subsequent time step $\tau + 1$. $\lambda_1$ and $\lambda_2$ are hyper-parameters for the regularization terms, $r_t$ represents the actual acceptance at time $t$, as depicted in Figure 2 (d), and $\bar{r}_t$ denotes the acceptance estimated by Equation 2.

$$\min_{\psi,\phi,\boldsymbol{\Theta},\boldsymbol{U}} \mathcal{L} = \sum_{t=1}^{\tau} \left|\bar{r}_t - r_t\right|^2 + \lambda_1 \left\|\boldsymbol{\Theta}\right\|^2 + \lambda_2 \left\|\boldsymbol{U}\right\|^2 \tag{4}$$

## 3.2. Negotiation UCB

In this subsection, building upon the above formulation, we introduce the NegUCB algorithm, a simple yet effective approach. In this algorithm, bids are represented as indicator vectors indicating the items involved in each bid. Please refer to § B.2 for detailed examples.

Since simultaneously deriving the functions $\phi$ and $\psi$, as well as the parameters $\boldsymbol{\Theta}$ and $\boldsymbol{U}$ is challenging, we assume the format of the function $\psi$ in Assumption 3.1. In § 3.2.1, we provide the closed-form solutions for $\boldsymbol{\Theta}$ and $\boldsymbol{U}$, which depend on function $\phi$. In § 3.2.2, we use kernel regression (Schulz et al., 2018) to eliminate the dependency on function $\phi$. At last, we summarize the NegUCB algorithm in § 3.2.3.

**Assumption 3.1.** If the contexts of items are characterized by their basic features, the extraction function $\psi$ in Equation 5 can accurately capture the context of bid $\boldsymbol{b}_\tau$. In other words, it encompasses substantial information about the items included in the bid.

$$\psi(\boldsymbol{Y}, \boldsymbol{b}_\tau) = \boldsymbol{Y}^\top \boldsymbol{b}_\tau^\top \tag{5}$$

Despite the linearity assumption on $\psi$, the acceptance function is non-linear because of the transforming function $\phi$.

### 3.2.1. Parameters

Obviously, the objective function $\mathcal{L}$ is not jointly convex concerning both $\boldsymbol{\Theta}$ and $\boldsymbol{U}$. However, it is convex concerning one parameter if the other one is fixed. Therefore, we employ an alternative least square optimization approach, iterating the calculation of one parameter with a closed-form solution while keeping the other parameter fixed. Based on Assumption 3.1, the closed-form solution for $\boldsymbol{\Theta}$ is as Equation 6, while that for $\boldsymbol{U}$ is as Equation 7.

$$\mathrm{vec}(\boldsymbol{\Theta}) = (\boldsymbol{A}_\tau^\top \boldsymbol{A}_\tau + \lambda_1 \boldsymbol{I}_{h^2})^{-1} \boldsymbol{A}_\tau^\top (\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U})) \tag{6}$$

$$\mathrm{vec}(\boldsymbol{U}) = (\boldsymbol{D}_\tau^\top \boldsymbol{D}_\tau + \lambda_2 \boldsymbol{I}_{mh})^{-1} \boldsymbol{D}_\tau^\top (\boldsymbol{r}_\tau - \boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta})) \tag{7}$$

Rows of matrices $\boldsymbol{A}_\tau$ and $\boldsymbol{D}_\tau$ are samples as $\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_t)$ and $\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \boldsymbol{p}_t$ where $\boldsymbol{p}_t \in R^{1\times m}$ is a one-hot vector representing the counterpart index at time step $t = 1, 2, ..., \tau$. It is evident that the solutions for parameters $\boldsymbol{\Theta}$ and $\boldsymbol{U}$ are contingent on the transformation function $\phi$, which can take various forms, such as polynomial functions, neural networks, etc., and thus needs to be learned.

### 3.2.2. Transformation function

Given the limited amount of negotiation data with various counterparts, learning $\phi$ becomes intractable if it involves many parameters, such as in the case of neural networks. In NegUCB, we utilize Reproducing Kernel Hilbert Spaces within kernel functions to avoid the need for learning $\phi$, enhancing efficiency. Moreover, since iterating among three components, i.e., learning $\phi$, $\boldsymbol{U}$, and $\boldsymbol{\Theta}$, is highly unstable, NegUCB iterates between learning $\boldsymbol{U}$ and $\boldsymbol{\Theta}$, significantly improving the learning stability.

Corresponding to matrices $\boldsymbol{A}_\tau$ and $\boldsymbol{D}_\tau$ dependent on function $\phi$, we define matrices $\boldsymbol{K}_\tau$ and $\boldsymbol{Z}_\tau$. Each entry $(\boldsymbol{K}_\tau)_{t,j}$ and $(\boldsymbol{Z}_\tau)_{t,j}$ are the dot product of the $t$-th and $j$-th samples of $\boldsymbol{A}_\tau$ and $\boldsymbol{D}_\tau$, respectively. By Assumption 3.2, we can calculate $\boldsymbol{K}_\tau$ and $\boldsymbol{Z}_\tau$ without knowing $\phi$.

**Assumption 3.2.** Each entry of $\boldsymbol{K}_\tau$ and $\boldsymbol{Z}_\tau$ can be calculated by Equation 8 and Equation 9 respectively, where $t, j = 1, 2, ..., \tau$, and $\kappa_1$ and $\kappa_2$ are two kernel functions.

$$(\boldsymbol{K}_\tau)_{t,j} = \kappa_1(\boldsymbol{x}_t, \boldsymbol{x}_j) \times \kappa_1(\boldsymbol{b}_t\boldsymbol{Y}, \boldsymbol{b}_j\boldsymbol{Y}) \tag{8}$$

$$(\boldsymbol{Z}_\tau)_{t,j} = \begin{cases} \kappa_2(\boldsymbol{b}_t\boldsymbol{Y}, \boldsymbol{b}_j\boldsymbol{Y}) & \boldsymbol{p}_t = \boldsymbol{p}_j \\ 0 & \boldsymbol{p}_t \neq \boldsymbol{p}_j \end{cases} \tag{9}$$

Denoting the above entry values as $k_{tj}$ and $z_{tj}$, then the kernel vectors at time step $\tau$ are $\boldsymbol{k}_\tau = (k_{1\tau}, k_{2\tau}, ..., k_{\tau,\tau})$ and $\boldsymbol{z}_\tau = (z_{1\tau}, z_{2\tau}, ..., z_{\tau,\tau})$, and $\boldsymbol{K}_\tau$ and $\boldsymbol{Z}_\tau$ are the kernel matrices. Based on Assumption 3.2, we have Lemma 3.3 to approximate the acceptance function.

**Lemma 3.3.** *Instead of learning transformation function $\phi$, parameters $\boldsymbol{\Theta}$ and $\boldsymbol{U}$, and then estimating $r_{\tau+1}(\boldsymbol{b})$ by Equation 2, it is equivalent to iterate Equation 10 and Equation 11, then estimate $r_{\tau+1}(\boldsymbol{b})$ using Equation 12. Specifically, $\bar{\boldsymbol{k}}_{\tau+1} = \boldsymbol{k}_{\tau+1}[1:\tau]$ and $\bar{\boldsymbol{z}}_{\tau+1} = \boldsymbol{z}_{\tau+1}[1:\tau]$, which are $\boldsymbol{k}_{\tau+1}$ and $\boldsymbol{z}_{\tau+1}$ without their last entries.*

$$\boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta}) = \boldsymbol{K}_\tau(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U})) \tag{10}$$

$$\boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U}) = \boldsymbol{Z}_\tau(\boldsymbol{Z}_\tau + \lambda_2 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta})) \tag{11}$$

$$\begin{aligned} \bar{r}_{\tau+1}(\boldsymbol{b}) = {} & \bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U})) \\ & + \bar{\boldsymbol{z}}_{\tau+1}(\boldsymbol{Z}_\tau + \lambda_2 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta})) \end{aligned} \tag{12}$$

Considering the definitions of $\boldsymbol{A}_\tau$ and $\boldsymbol{D}_\tau$, it is evident that $\boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta})$ and $\boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U})$ are partial acceptances corresponding to the two terms in Equation 2 for historical time steps $t = 1, 2, ..., \tau$. From the iteration results, the two terms in Equation 12 estimate the respective terms in Equation 2 for the subsequent time step $\tau + 1$.

#### 3.2.3. NegUCB Algorithm

For the subsequent time step $\tau+1$, we can estimate $r_{\tau+1}(\boldsymbol{b})$ for each bid $\boldsymbol{b} \in B_{\tau+1}$ using Equation 12, then choose a bid to put forth or decide to accept or reject the bid from the counterpart by Equation 1. However, this approach relies solely on exploiting historical data, which may lead to sub-optimal choices. Hence, we explore the estimation uncertainty based on the Upper Confidence Bound principle (Li et al., 2010; Valko et al., 2013; Liu et al., 2018).

Instead of Equation 1, we make decisions by Equation 13, where $e_{\tau+1}$ measures the estimation variance and is expressed in Equation 14. Parameters $\alpha_\theta$ and $\alpha_u$ are elaborated in Lemma 3.4. For notation conciseness, we use $k_{\tau+1}$ and $z_{\tau+1}$ to denote $k_{\tau+1,\tau+1}$ and $z_{\tau+1,\tau+1}$.

$$\boldsymbol{b}^*_{\tau+1} = \arg\max\{\bar{r}_{\tau+1}(\boldsymbol{b}) + e_{\tau+1}(\boldsymbol{b})\} \times f_{\tau+1}(\boldsymbol{b}) \tag{13}$$

$$\begin{aligned} e_{\tau+1}(\boldsymbol{b}) = &\frac{\alpha_\theta}{\sqrt{\lambda_1}}\sqrt{k_{\tau+1} - \bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}\bar{\boldsymbol{k}}^\top_{\tau+1}} \\ &+ \frac{\alpha_u}{\sqrt{\lambda_2}}\sqrt{z_{\tau+1} - \bar{\boldsymbol{z}}_{\tau+1}(\boldsymbol{Z}_\tau + \lambda_2 \boldsymbol{I}_\tau)^{-1}\bar{\boldsymbol{z}}^\top_{\tau+1}} \end{aligned} \tag{14}$$

Integrating exploitation and exploration, NegUCB is implemented online as Algorithm 1, where we use $\boldsymbol{a}_\tau$ and $\boldsymbol{d}_\tau$ to respectively denote $\boldsymbol{A}_\tau \mathrm{vec}(\boldsymbol{\Theta})$ and $\boldsymbol{D}_\tau \mathrm{vec}(\boldsymbol{U})$ for notation conciseness. *Online* means the parameters are incrementally updated each time new negotiation data is generated. NegUCB essentially iterates between **Step 1.** Estimating the second term in Equation 2, then calculating the first term; **Step 2.** Estimating the first term in Equation 2, then calculating the second term.

**Algorithm 1** NegUCB Algorithm

```
Input: $\lambda_1, \lambda_2 \in (0, +\infty)$, kernel functions $\kappa_1$, $\kappa_2$
Output: vectors $\boldsymbol{a}_N$ and $\boldsymbol{d}_N$
for $\tau = 1$ to $N$ do
  select bid $\boldsymbol{b}_\tau$ randomly if $\tau = 1$, or according to Equation 13 if $\tau > 1$, and observe $r_\tau$
  if $\tau = 1$ then
    initialize $\boldsymbol{d}_{\tau-1}$ as an empty vector
    initialize kernel matrix $\boldsymbol{Z}_\tau = [z_\tau]$ and set $a_\tau = r_\tau$
  else
    kernel matrix $\boldsymbol{Z}_\tau = [\boldsymbol{Z}_{\tau-1}, \bar{\boldsymbol{z}}^\top_\tau; \bar{\boldsymbol{z}}_\tau, z_\tau]$
    calculate $a_\tau = r_\tau - \bar{\boldsymbol{z}}_\tau(\boldsymbol{Z}_{\tau-1} + \lambda_2 \boldsymbol{I}_{\tau-1})^{-1}\boldsymbol{d}_{\tau-1}$
  end if
  if $\tau = 1$ then
    initialize kernel matrix $\boldsymbol{K}_\tau = [k_\tau]$ and $\boldsymbol{a}_\tau = (a_\tau)$
  else
    kernel matrix $\boldsymbol{K}_\tau = [\boldsymbol{K}_{\tau-1}, \bar{\boldsymbol{k}}^\top_\tau; \bar{\boldsymbol{k}}_\tau, k_\tau]$
    $\boldsymbol{a}_\tau = (\boldsymbol{a}_{\tau-1}; a_\tau)$
  end if
  calculate $d_\tau = r_\tau - \boldsymbol{k}_\tau(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}\boldsymbol{a}_\tau$
  $\boldsymbol{d}_\tau = (\boldsymbol{d}_{\tau-1}; d_\tau)$
end for
```

### 3.3. Theoretical Analysis to NegUCB

**Lemma 3.4.** *If the true parameters satisfy $\|\boldsymbol{\Theta}_*\| \le \beta_\theta$ and $\|\boldsymbol{U}_*\| \le \beta_u$, the samples satisfy $\|\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_t)\| \le 1$ and $\|\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \boldsymbol{p}_t\| \le 1$ for $t = 1, 2, ..., \tau$, then with probability at least $1 - \sqrt{\delta}$, the two terms in Equation 12 have estimation error bounds $\alpha_\theta$ and $\alpha_u$ as follows. Here $h_*$ and $m_*$ are the effective dimensions of $\boldsymbol{\mathcal{A}}_\tau = \boldsymbol{A}^\top_\tau \boldsymbol{A}_\tau + \lambda_1 \boldsymbol{I}_{h^2}$ and $\boldsymbol{\mathcal{D}}_\tau = \boldsymbol{D}^\top_\tau \boldsymbol{D}_\tau + \lambda_2 \boldsymbol{I}_{mh}$, $p, q \in (0, 1)$ are constants.*

$$\begin{aligned} \alpha_\theta =& \|\mathrm{vec}(\boldsymbol{\Theta}_\tau) - \mathrm{vec}(\boldsymbol{\Theta}_*)\|_{\boldsymbol{\mathcal{A}}_\tau} \\ \le& \lambda_1\beta_\theta + \sqrt{h_* \mathrm{log}(1 + \frac{\tau}{\lambda_1 h_*}) - \mathrm{log}\delta} + \frac{2\beta_u}{\sqrt{\lambda_1 q}} \end{aligned} \tag{15}$$

$$\begin{aligned} \alpha_u =& \|\mathrm{vec}(\boldsymbol{U}_\tau) - \mathrm{vec}(\boldsymbol{U}_*)\|_{\boldsymbol{\mathcal{D}}_\tau} \\ \le& \lambda_2\beta_u + \sqrt{m_* \mathrm{log}(1 + \frac{\tau}{\lambda_2 m_*}) - \mathrm{log}\delta} + \frac{2\beta_\theta}{\sqrt{\lambda_2 p}} \end{aligned} \tag{16}$$

In Lemma 3.4, $\boldsymbol{\mathcal{A}}_\tau$ and $\boldsymbol{\mathcal{D}}_\tau$ correspond to the first item of Equation 6 and Equation 7, respectively. *Effective dimension* (Valko et al., 2013; Vakili et al., 2021) is a commonly used concept in kernel regression and can be considered as the number of principal dimensions. They contract the bounds as $h_* \ll h^2$ and $m_* \ll mh$, where $h$ is the dimension of $\mathcal{H}$. Bounds of each sample $\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_t)$ and $\phi(\boldsymbol{b}_t\boldsymbol{Y}) \otimes \boldsymbol{p}_t$ are set as 1 for description convenience. They correlate with the number of items in the bid, referred to as the *bid cardinality* and denoted as $\gamma \leqslant n \in \mathbb{Z}^+$. We can guarantee the bounds of samples by normalizing the contexts $\boldsymbol{X}$ and $\boldsymbol{Y}$. Based on Lemma 3.4, we guarantee the performance of NegUCB by the following theorem.

**Theorem 3.5.** *Under the same assumptions as Lemma 3.4, with probability at least $1 - \sqrt{\delta}$, the cumulative regret of Algorithm 1 has the following upper bound, where $r_t(\boldsymbol{b}^*_t)$ and $r_t(\boldsymbol{b}_t)$ are respectively the true acceptance of the optimal bid $\mathbf{b}^*_t$ and the bid chosen by Algorithm 1 at time step $t$. $\alpha_f$ is the union bound of the benefit functions, i.e., $|f_t(\boldsymbol{b})| \le \alpha_f$ for $\forall \boldsymbol{b} \in B_t$ and $\forall t \in \{1, 2, ..., \tau\}$.*

$$\begin{aligned} &\sum_{t=0}^{\tau} r_t(\boldsymbol{b}^*_t) \times f_t(\boldsymbol{b}^*_t) - r_t(\boldsymbol{b}_t) \times f_t(\boldsymbol{b}_t) \\ \le& 2\alpha_\theta\alpha_f\sqrt{2h_*\tau\mathrm{log}(1 + \frac{\tau}{\lambda_1 h_*})} \\ &+ 2\alpha_u\alpha_f\sqrt{2m_*\tau\,\mathrm{log}(1 + \frac{\tau}{\lambda_2 m_*})} \end{aligned} \tag{17}$$

Indeed, the cumulative regret is sub-linear concerning the number of time steps $\tau$. This implies that as the number

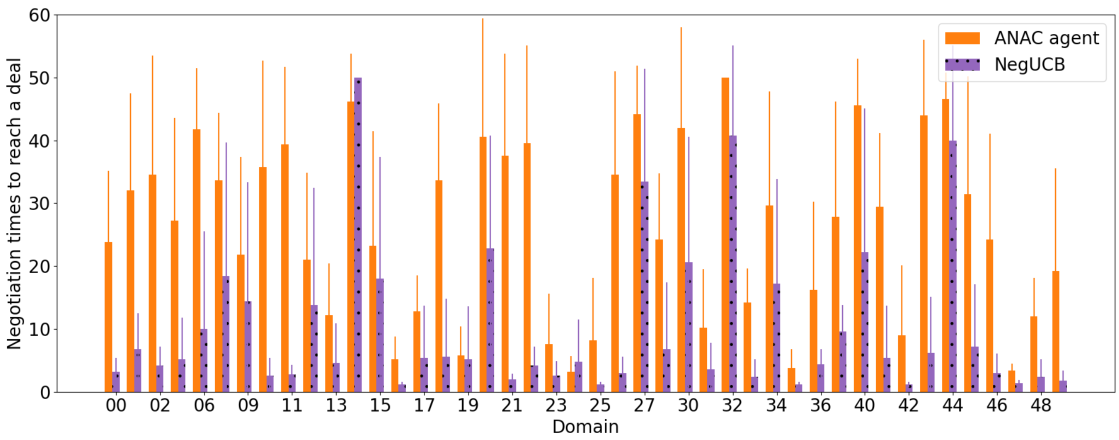


*Figure 3.* Negotiation steps needed to reach a deal on each ANAC domain of *domain 00 - 49*.

of negotiation steps increases, the cumulative regret grows at a slower rate, indicating improved negotiation capability. Besides, the bound is independent of the bid cardinality $\gamma$, distinguishing NegUCB from existing works (Qin et al., 2014; Wen et al., 2015; Chen et al., 2018; Nie et al., 2022; Fourati et al., 2023). It is a result of the full-bandit feedback and Assumption 3.1. The effect from bid cardinality to the cumulative regret bound is further discussed in § A.4.1.

## 4. Experiments

In this section, we evaluate NegUCB across the three representative negotiation tasks depicted in Figure 1, comparing it with five representative baselines: ANAC agent [1], LinUCB, FactorUCB, KernelUCB, and a reinforcement learning-based negotiation method (Cao et al., 2018; Bagga et al., 2020). It is important to note that we extend the original UCB-based baselines to handle combinatorial bandits and full-bandit feedback effectively. Further analysis regarding the rationale behind baseline selection is in § B.1.

As mentioned, the benefit function $f_\tau$ is problem-specific. In our experiments, we set $f_\tau(\boldsymbol{b}_\tau) = 1$ if $\boldsymbol{b}_\tau \in \mathcal{C} \cap B_\tau$, where $\mathcal{C}$ consists of bids satisfying certain beneficial constraints, otherwise, $f_\tau(\boldsymbol{b}_\tau) = 0$. It implies that we encourage bids that are advantageous to us. This simple configuration lets us concentrate on the acceptance function $r_\tau$ rather than the handcrafted $f_\tau$. The subsections of specific tasks will further define the set $\mathcal{C}$ constraining bids.

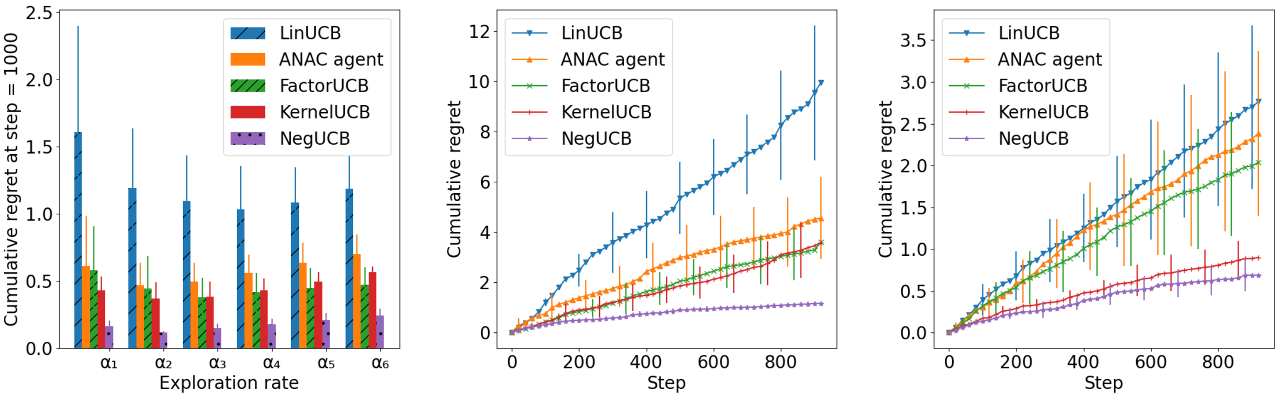
### 4.1. Multi-issue Negotiation

ANAC (Automated Negotiating Agents Competition) is an international tournament that has been held since 2010, providing 50 negotiation domains. However, compared to the settings of NegUCB, ANAC tasks are relatively simple. For instance, negotiators and items lack contexts, and there is only one negotiator pair for each domain. Consequently, some components of NegUCB are not necessary for these tasks. In this subsection, we modify NegUCB for compatibility with ANAC tasks, showing the adaptability of NegUCB to diverse negotiation problems. In ANAC experiment, NegUCB does not consider any context and relies on inferring hidden states of negotiators and items from negotiation experiences. Essentially, it degenerates into traditional combinatorial bandits. In contrast, most of the ANAC agents submitted by tournament participants, including the winners (Aydogan et al., 2023), are rule-based.

Original ANAC tasks impose a strict deadline on negotiation, limiting each negotiation pair to a constant number of negotiation steps. Negotiators are aware of this deadline and can strategically utilize it. This setup diverges from our setting in that a negotiator may lose patience at any time, and its counterpart may not be aware of it. Therefore, in this subsection, we redefine the task. First, we eliminate the deadline and investigate the number of rounds needed to reach a deal. Second, we define the constraining set $\mathcal{C}$ only contains bids whose utilities are larger than the mean utility of all possible bids to our negotiator. Agents achieving a deal in fewer steps are more effective. Based on the insights of the ANAC agents submitted by tournament participants, we modify them to be compatible with the redefined task. Specifically, the ANAC agent we adopt in this experiment randomly selects a valid bid from those with the highest utility rankings for our negotiator.

We investigate the number of negotiation steps required to achieve a deal by each algorithm across 50 ANAC domains, specifically from *domain 00* to *domain 49*. For *domain 3*, *domain 4*, *domain 7*, *domain 28*, *domain 37*, and *domain 38*, both algorithms failed to reach a deal in 50 rounds, thus for the sake of conciseness, we show the results on the remaining 44 domains in Figure 3. NegUCB consistently achieves beneficial deals much earlier across almost

[1] http://ii.tudelft.nl/nego/node/7



(a) × 1000: Cumulative theoretical regret (b) × 100: Cumulative theoretical regret (c) × 100: Cumulative acceptance regret

*Figure 4.* Experiment results of resource allocation task. *Theoretical regret* represents the difference between the estimated $\bar{r}$ and the simulated $r$. *Acceptance regret* refers to the difference between the estimated and simulated acceptance.

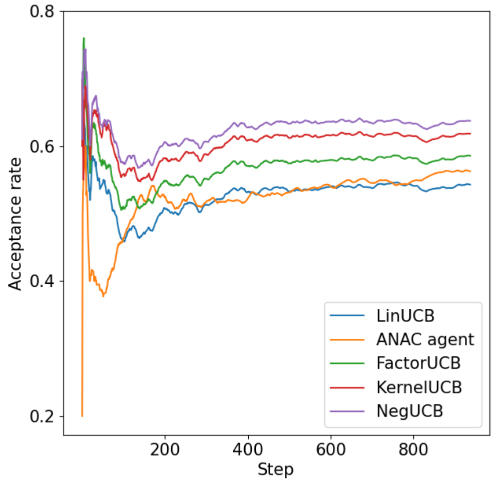


*Figure 5.* Acceptance rate on resource allocation task, which is defined as the percentage of the proposed bids being accepted.

all ANAC domains. Considering the effectiveness of the simplified NegUCB used in this subsection, it is adopted in place of the ANAC agent in the following experiments for a more appropriate comparison.

Additionally, we analyze the action spaces. Considering *domain 13* for example, it has $4$ issues, each of which has $6, 12, 5, 26$ possible values to choose from, then the bid set contains at most $6 \times 12 \times 5 \times 26 = 9360$ choices. Similarly, other domains exhibit comparable action space sizes.

### 4.2. Resource Allocation

Motivated by experiments of existing works (Cao et al., 2018), we design a resource allocation task.

Assume there are three categories of items, and the number of items in each category does not exceed $5$. Each item category has a randomly generated context vector denoted as $\boldsymbol{y}_j, j = 1, 2, 3$. A context vector $\boldsymbol{x}_w$ and a hidden state vector $\boldsymbol{u}_w$ are randomly generated for each of the 30 negotiator pairs. For simplicity, we assume that $\boldsymbol{x}_w$, $\boldsymbol{y}_j$, and $\boldsymbol{u}_w$ are all 2-dimensional, with each entry in the range $[0, 1]$. The acceptance function is simulated using Equation 2, where the transformation function is as Equation 18. Besides, we draw the parameter matrix $\boldsymbol{\Theta}$ of size $6 \times 6$ from a Gaussian distribution $\mathcal{N}(0, 1)$. The counterpart accepts the bid if the simulated acceptance $r$ satisfies $r > 0$.

$$\phi(\boldsymbol{x}) = (\frac{1}{\sqrt{2}}, x_1, x_2, \frac{1}{\sqrt{2}}x_1^2, x_1x_2, \frac{1}{\sqrt{2}}x_2^2) \tag{18}$$

Specifically, the transformation function is the basis function of polynomial kernel $\kappa(\boldsymbol{x}_w, \boldsymbol{x}_j) = \frac{1}{2}(\boldsymbol{x}_w\boldsymbol{x}_j^\top + 1)^2$. Furthermore, we define the set $\mathcal{C}$ contains bids that allow our negotiator to acquire more items than the counterpart.

Figure 4(a) shows the cumulative theoretical regret for each algorithm under various exploration parameters, i.e., $\alpha_\theta = \alpha_u = \alpha_1, \alpha_2, ..., \alpha_6$ summarized in § B.3. It is evident that the cumulative theoretical regret for each algorithm decreases initially and then increases, illustrating the advantages of exploration and the drawbacks of over-exploration. Figure 4(b) and Figure 4(c) display the cumulative theoretical regret and cumulative acceptance regret of each algorithm at each time step under their corresponding optimal exploration parameter, respectively. Figure 5 illustrates the acceptance rate of each algorithm under their corresponding optimal exploration parameter. Given the random nature of exploration in reinforcement learning, i.e., $\epsilon$-greedy, we extend the training duration of the reinforcement learning method to 20000 steps to ensure the results accurately reflect its true capabilities. Its final result reaches an acceptance rate lower than $0.6$. Refer to § B.3 for a detailed insight into its convergence process. From these results, we can observe clear advantages of NegUCB.

In this experiment, the action space comprises at most $6 \times 6 \times 6 = 216$ actions, which aligns with that in the existing work (Cao et al., 2018) to confirm the effectiveness of NegUCB for established problems. We add another experiment with a larger action space in Appendix B.3.

### 4.3. Trading

CivRealm (Qi et al., 2024) is an interactive environment designed for the open-source strategy game *Freeciv*. In this environment, multiple players engage in their civilizations' simultaneous development and competition. Alongside elements such as land, population, and economy, each player possesses a technology tree, allowing them to research and acquire the 87 technologies progressively.

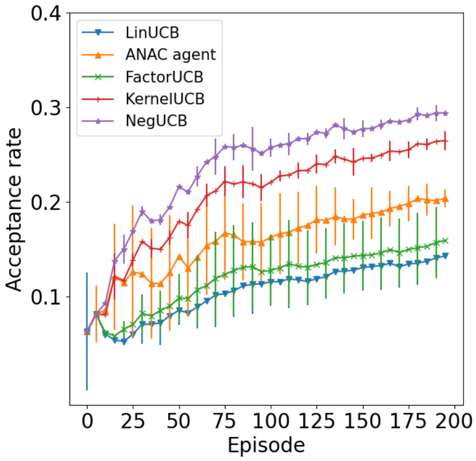
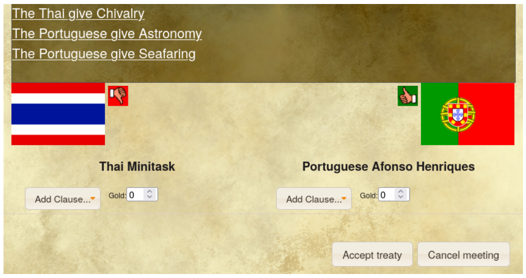


*Figure 6.* Acceptance rate at each episode on trading task

One crucial feature of CivRealm is its *Diplomacy* component, enabling players to engage in technology trades. For instance, if our negotiator possesses the technology *Chivalry* and seeks the technology *Astronomy*, besides researching it by itself, our negotiator can also acquire it through trading with other players who already possess *Astronomy*. A negotiation window of CivRealm is as Figure 7 [2]. A negotiator can counter-propose or cancel the meeting if they reject the bid. On the other hand, the negotiator can accept the bid by accepting the treaty. In this experiment, $\boldsymbol{b}_\tau \in \mathcal{C}$ if the total cost of the given technologies is no more than that of the required ones. For practical reasons, we set the bid cardinality as $\gamma = 4$, with details explained in § B.4.

In this experiment, we systematically explore SE kernels with diverse hyper-parameters $\sigma$ to fine-tune the most suitable kernel function for the technology trading task in CivRealm. Based on the results, we conclude that the SE kernel with $\sigma = 1$ emerges as the most suitable choice for this task. Besides, we tune the exploration rate ranging from 0 to 1 and choose 0.1 as the optimal exploration rate for NegUCB. Please refer to § B.4 for more details. Surprisingly, apart from the baselines LinUCB, KernelUCB, and our proposed method NegUCB, other baselines fail to demonstrate improvements with increased exploration. We attribute this observation to the complexity of the task, where inaccurate formulations result in misguided exploration strategies. Figure 6 illustrates the acceptance rates of each algorithm under their corresponding optimal exploration parameters, i.e., $0.1, 0, 0, 0.1, 0.1$, affirming the clear advantages of NegUCB. The reinforcement learning method is not utilized in this experiment due to the challenge associated with handling such a large action space, whose cardinality is at most $\sum_{j=1}^{\gamma} \binom{87}{j}$.

**Case Study.** A negotiation case on CivRealm is depicted in Figure 7. Thailand proposed to give *Chivalry* and seek *Astronomy* and *Seafaring* from Portugal with costs of 270, 185, and 112, respectively. The net income for Portugal would be $270 - 185 - 112 = -27$. However, according to the running game, Portugal accepted the bid. It suggests the presence of hidden states that we did not observe, influencing Portugal's decision to accept the bid. Without the hidden state component in NegUCB, we might overlook such bids, substantiating that hidden states play a crucial role in estimating the counterpart's decisions.


The Thai give Chivalry
The Portuguese give Astronomy
The Portuguese give Seafaring

Thai Minitask

Portuguese Afonso Henriques

*Figure 7.* A case of negotiation in CivRealm. Thailand is our negotiator, while Portugal is our counterpart.

## 5. Conclusion

This paper introduces a comprehensive formulation for negotiation, grounded in contextual combinatorial multi-armed bandits, capable of encompassing a broad spectrum of real-world negotiation tasks. Building upon this formulation, we propose the NegUCB algorithm as a solution to address the four prevalent challenges in negotiation: the exploitation-exploration dilemma, handling large action spaces, partial observations, and complex acceptance functions. Under mild assumptions, NegUCB ensures a regret upper bound that is sub-linear with respect to the negotiation steps and independent of the bid cardinality. A series of experiments on diverse negotiation tasks validate NegUCB's effectiveness and advantages in learning negotiation strategies.

[2] It is from a running Freeciv game and may contain politically sensitive names of nations, which are purely hypothetical.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China.

## Impact Statement

This paper aims to advance the field of negotiation among agents with diverse interests from a multi-armed bandit perspective, which is well-suited for negotiation problems and has been well-investigated. It encourages future research to tackle any limitation of our method under the general formulation utilizing the advantages of bandit-based techniques. The model applies to scenarios where the agent knows the negotiation target it is trying to reach with the counterpart at each time step, which is a very mild constraint in *negotiation*. However, for tasks where hypermetropic *planning* plays the main role and negotiation is only one type of action that merely assists the agent in completing the task, i.e., those fall in the reinforcement learning paradigm, the model presented in this work may be limited due to the lack of an explicitly given negotiation target.

## References

Agarwal, M., Aggarwal, V., Umrawal, A. K., and Quinn, C. Dart: Adaptive accept reject algorithm for non-linear combinatorial bandits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 6557–6565, 2021.

Aydogan, R., Baarslag, T., Fujita, K., Hoos, H. H., Jonker, C. M., Mohammad, Y., and Renting, B. M. The 13th international automated negotiating agent competition challenges and results. Technical report, In International Joint Conference on Artificial Intelligence, 2023.

Bagga, P., Paoletti, N., Alrayes, B., and Stathis, K. A deep reinforcement learning approach to concurrent bilateral negotiation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 297–303, 2020.

Bakker, J., Hammond, A., Bloembergen, D., and Baarslag, T. Rlboa: A modular reinforcement learning framework for autonomous negotiating agents. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pp. 260–268, 2019.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural actor–critic algorithms. *Automatica*, 45(11): 2471–2482, 2009.

Buron, C. L. R., Guessoum, Z., and Ductor, S. Mcts-based automated negotiation agent. In *Proceedings of the 22nd International Conference on Principles and Practice of Multi-Agent System*, 2019.

Cao, K., Lazaridou, A., Lanctot, M., Leibo, J. Z., Tuyls, K., and Clark, S. Emergent communication through negotiation. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Chen, L., Xu, J., and Lu, Z. Contextual combinatorial multi-armed bandits with volatile arms and submodular reward. In *Advances in Neural Information Processing Systems*, 2018.

Chen, W., Wang, Y., and Yuan, Y. Combinatorial multi-armed bandit: General framework, results and applications. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 151–159. PMLR, 2013.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dulac-Arnold, G., Evans, R., v. Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679.*, 2016.

FAIR, M. F. A. R. D. T., Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Fourati, F., Aggarwal, V., Quinn, C. J., and Alouini, M. S. Randomized greedy learning for non-monotone stochastic submodular maximization under full-bandit feedback. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistic*, pp. 7455–7471. PMLR, 2023.

Haasdonk, B. and Pekalska, E. Classification with kernel mahalanobis distance classifiers. In *Advances in Data Analysis, Data Handling and Business Intelligence*, pp. 351–361, 2010.

Hwang, T., Chai, K., and Oh, M. Combinatorial neural bandits. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 14203–14236. PMLR, 2023.

Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., and Batra, D. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125.*, 2017.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 661–670, 2010.

Liu, B., Wei, Y., Zhang, Y., Yan, Z., and Yang, Q. Transferable contextual bandit for cross-domain recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Liu, T. and Zheng, Z. Negotiation assistant bot of pricing prediction based on machine learning. *International Journal of Intelligence Science*, 10(02), 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., D. Wierstra1, S. L., and Hassabis, D. Human-level control through deep reinforcement learning. *nature*, pp. 529–533, 2015.

Nie, G., Agarwal, M., Umrawal, A. K., Aggarwal, V., and Quinn, C. J. An explore-then-commit algorithm for submodular maximization under full-bandit feedback. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, pp. 1541–1551. PMLR, 2022.

Paquette, P., Lu, Y., Bocco, S., Smith, M. O., O-G, S., Kummerfeld, J. K., Singh, S., Pineau, J., and Courville, A. No-press diplomacy: Modeling multi-agent gameplay. In *Advances in Neural Information Processing Systems*, 2019.

Qi, S., Chen, S., Li, Y., Kong, X., Wang, J., Yang, B., Wong, P., Zhong, Y., Zhang, X., Zhang, Z., Liu, N., Wang, W., Yang, Y., and Zhu, S. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. In *Proceedings of the 12nd International Conference on Learning Representations*, 2024.

Qin, L., Chen, S., and Zhu, X. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 461–469, 2014.

Rejwan, I. and Mansour, Y. Top-k combinatorial bandits with full-bandit feedback. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, pp. 752–776. PMLR, 2020.

Rodriguez-Fernandez, J., Pinto, T., Silva, F., Praça, I., Vale, Z., and Corchado, J. Context aware q-learning-based model for decision support in the negotiation of energy contracts. *International Journal of Electrical Power & Energy Systems*, pp. 489–501, 2019.

Schulz, E., Speekenbrink, M., and Krause, A. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

Sengupta, A., Nakadai, S., and Mohammad, Y. Transfer learning based adaptive automated negotiating agent framework. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pp. 468–474, 2022.

Uschmajew, A. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 33 (2):639–652, 2012.

Vakili, S., Khezeli, K., and Picheny, V. On information gain and regret bounds in gaussian process bandits. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pp. 82–90. PMLR, 2021.

Vakili, S., Ahmed, D., Bernacchia, A., and Pike-Burke, C. Delayed feedback in kernel bandits. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv: 1309.6869.*, 2013.

Wang, H., Wu, Q., and Wang, H. Factorization bandits for interactive recommendation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Wen, Z., Kveton, B., and Ashkan, A. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1113–1122. PMLR, 2015.

Xu, P., Wen, Z., Zhao, H., and Gu, Q. Neural contextual bandits with deep representation and shallow exploration. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Yadkori, Y. A., Pal, D., and Szepesvari, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, 2011.

Zhou, D., Li, L., and Gu, Q. Neural contextual bandits with ucb-based exploration. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

## A. Proofs

In this section, we first provide a derivation of the closed-form solutions in Appendix A.1, then we provide the proofs of Lemma 3.3, Lemma 3.4 and Theorem 3.5 in Appendix A.2, Appendix A.3, and Appendix A.4, respectively.

### A.1. Derivation of Closed-form Solutions

Under Assumption 3.1, the approximated acceptance function and objective function are respectively:

$$
\begin{aligned}
&\bar{r}_\tau(\boldsymbol{b}_\tau) = \phi(\boldsymbol{x}_\tau)\boldsymbol{\Theta}\phi(\boldsymbol{Y}^\top \boldsymbol{b}_\tau^\top) + \boldsymbol{p}_\tau \boldsymbol{U}\phi(\boldsymbol{Y}^\top \boldsymbol{b}_\tau^\top) \\
&\mathcal{L} = \sum_{t=1}^{\tau} \left| \phi(\boldsymbol{x}_t)\boldsymbol{\Theta}\phi(\boldsymbol{Y}^\top \boldsymbol{b}_t^\top) + \boldsymbol{p}_t \boldsymbol{U}\phi(\boldsymbol{Y}^\top \boldsymbol{b}_t^\top) - r_t \right|^2 + \lambda_1 \left\| \boldsymbol{\Theta} \right\|^2 + \lambda_2 \left\| \boldsymbol{U} \right\|^2
\end{aligned}
$$

Based on basic linear algebra, we can derive the closed-form solutions of $\boldsymbol{\Theta}$ and $\boldsymbol{U}$ easily (Li et al., 2010; Wang et al., 2017). The core method we employ in the derivation relies on the conclusion that $\boldsymbol{a}\boldsymbol{B}\boldsymbol{c}^\top = (\boldsymbol{c} \otimes \boldsymbol{a})\text{vec}(\boldsymbol{B})$, where $\boldsymbol{a}$ and $\boldsymbol{c}$ denote any two row-vectors, and $\boldsymbol{B}$ denotes any matrix, provided that their sizes match.

### A.2. Proof of Lemma 3.3

Proof in this subsection directly uses the closed-form solutions in Equation 6 and Equation 7.

*Proof.* According to the closed-form solution in Equation 6, we have the following equation.

$$
\begin{aligned}
(\boldsymbol{A}_\tau^\top \boldsymbol{A}_\tau + \lambda_1 \boldsymbol{I}_{h^2})\text{vec}(\boldsymbol{\Theta}) &= \boldsymbol{A}_\tau^\top (\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U})) \\
\Rightarrow \text{vec}(\boldsymbol{\Theta}) &= \frac{1}{\lambda_1}(\boldsymbol{A}_\tau^\top (\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U})) - \boldsymbol{A}_\tau^\top \boldsymbol{A}_\tau \text{vec}(\boldsymbol{\Theta})) \\
&= \frac{1}{\lambda_1}\boldsymbol{A}_\tau^\top (\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U}) - \boldsymbol{A}_\tau \text{vec}(\boldsymbol{\Theta}))
\end{aligned}
$$

Denote $\boldsymbol{\alpha} = \frac{1}{\lambda_1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U}) - \boldsymbol{A}_\tau \text{vec}(\boldsymbol{\Theta}))$, thus there is $\text{vec}(\boldsymbol{\Theta}) = \boldsymbol{A}_\tau^\top \boldsymbol{\alpha}$. Integrating these two equations:

$$
\begin{aligned}
&\Rightarrow \boldsymbol{\alpha} = \frac{1}{\lambda_1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U}) - \boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top \boldsymbol{\alpha}) \\
&\Rightarrow \boldsymbol{\alpha} = (\boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U})) \\
&\Rightarrow \text{vec}(\boldsymbol{\Theta}) = \boldsymbol{A}_\tau^\top \boldsymbol{\alpha} = \boldsymbol{A}_\tau^\top (\boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U}))
\end{aligned}
$$

From the definition of rows of matrix $\boldsymbol{A}_\tau$, we have:

$$
\begin{aligned}
(\boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top)_{tj} =& (\phi(\boldsymbol{b}_t \boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_t))(\phi(\boldsymbol{b}_j \boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_j))^\top \\
=& (\phi(\boldsymbol{b}_t \boldsymbol{Y})\phi(\boldsymbol{b}_j \boldsymbol{Y})^\top) \times (\phi(\boldsymbol{x}_t)\phi(\boldsymbol{x}_j)^\top) \\
=& \kappa_1(\boldsymbol{b}_t \boldsymbol{Y}, \boldsymbol{b}_j \boldsymbol{Y}) \times \kappa_1(\boldsymbol{x}_t, \boldsymbol{x}_j) = (\boldsymbol{K}_\tau)_{tj}
\end{aligned}
$$

The second equality above is from the fact that any row vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2$ satisfy $(\boldsymbol{v}_1 \otimes \boldsymbol{\nu}_1)(\boldsymbol{v}_2 \otimes \boldsymbol{\nu}_2)^\top = (\boldsymbol{v}_1 \boldsymbol{v}_2^\top)(\boldsymbol{\nu}_1 \boldsymbol{\nu}_2^\top)$. As a result, we can derive $\boldsymbol{A}_\tau \text{vec}(\boldsymbol{\Theta})$ as follows.

$$
\begin{aligned}
\boldsymbol{A}_\tau \text{vec}(\boldsymbol{\Theta}) =& \boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top (\boldsymbol{A}_\tau \boldsymbol{A}_\tau^\top + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U})) \\
=& \boldsymbol{K}_\tau (\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau - \boldsymbol{D}_\tau \text{vec}(\boldsymbol{U}))
\end{aligned}
$$

For the next time step $\tau + 1$, there is:

$$\begin{aligned}\phi(\boldsymbol{x}_{\tau+1})\boldsymbol{\Theta}\phi(\boldsymbol{Y}^\top\boldsymbol{b}_{\tau+1}^\top) =&(\phi(\boldsymbol{b}_{\tau+1}\boldsymbol{Y})\otimes\phi(\boldsymbol{x}_{\tau+1}))\text{vec}(\boldsymbol{\Theta})\\ =&\bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau+\lambda_1\boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau-\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}))\end{aligned}$$

Derivation of $\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U})$ is similar, thus we omit it.

□

### A.3. Proof of Lemma 3.4

Denote $\boldsymbol{\mathcal{A}}_\tau = \boldsymbol{A}_\tau^\top\boldsymbol{A}_\tau + \lambda_1\boldsymbol{I}_{h^2}$ and $\boldsymbol{\mathcal{D}}_\tau = \boldsymbol{D}_\tau^\top\boldsymbol{D}_\tau + \lambda_2\boldsymbol{I}_{mh}$. $\boldsymbol{\Theta}_*$ is the true parameter while $\boldsymbol{\Theta}_\tau$ is the parameter estimated at time step $\tau$. $\phi(\boldsymbol{b}_t\boldsymbol{Y})\otimes\phi(\boldsymbol{x}_t)$ is the sample at time step $t = 1, 2, ..., \tau$.

*Proof.* The error of the estimated partial acceptance based on contexts corresponding to $\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y})\otimes\phi(\boldsymbol{x}_{t+1})$ is as follows.

$$\begin{aligned}&\left|\bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau+\lambda_1\boldsymbol{I}_\tau)^{-1}(\boldsymbol{r}_\tau-\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}))-(\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y})\otimes\phi(\boldsymbol{x}_{t+1}))\text{vec}(\boldsymbol{\Theta}_*)\right|\\ \leq&\left\|\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y})\otimes\phi(\boldsymbol{x}_{t+1})\right\|\left\|\text{vec}(\boldsymbol{\Theta}_\tau)-\text{vec}(\boldsymbol{\Theta}_*)\right\|_{\boldsymbol{\mathcal{A}}_\tau}\\ \leq&\left\|\text{vec}(\boldsymbol{\Theta}_\tau)-\text{vec}(\boldsymbol{\Theta}_*)\right\|_{\boldsymbol{\mathcal{A}}_\tau}\\ =&\left\|\boldsymbol{\mathcal{A}}_\tau(\text{vec}(\boldsymbol{\Theta}_\tau)-\text{vec}(\boldsymbol{\Theta}_*))\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\\ =&\left\|\boldsymbol{A}_\tau^\top(\boldsymbol{r}_\tau-\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}_{\tau-1}))-(\boldsymbol{A}_\tau^\top\boldsymbol{A}_\tau+\lambda_1\boldsymbol{I}_{h^2})\text{vec}(\boldsymbol{\Theta}_*)\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\\ =&\left\|\boldsymbol{A}_\tau^\top(\boldsymbol{r}_\tau-\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}_{\tau-1})-\boldsymbol{A}_\tau\text{vec}(\boldsymbol{\Theta}_*))-\lambda_1\text{vec}(\boldsymbol{\Theta}_*)\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\\ =&\left\|\boldsymbol{A}_\tau^\top\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}_*)-\boldsymbol{A}_\tau^\top\boldsymbol{D}_\tau\text{vec}(\boldsymbol{U}_{\tau-1})+\boldsymbol{A}_\tau^\top\boldsymbol{\epsilon}_\tau-\lambda_1\text{vec}(\boldsymbol{\Theta}_*)\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\\ \leq&\left\|\boldsymbol{A}_\tau^\top\boldsymbol{D}_\tau(\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_{\tau-1}))\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}+\left\|\boldsymbol{A}_\tau^\top\boldsymbol{\epsilon}_\tau\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}+\lambda_1\left\|\boldsymbol{\Theta}_*\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\end{aligned}$$

The first inequality holds when the minimum eigenvalue of $\boldsymbol{\mathcal{A}}_\tau$ is at least 1. The term $\epsilon_\tau$ accounts for sub-Gaussian noise to the acceptance function. Now, consider the first term above:

$$\begin{aligned}&\left\|\boldsymbol{A}_\tau^\top\boldsymbol{D}_\tau(\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_{\tau-1}))\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}}\\ \leq&\frac{1}{\sqrt{\lambda_1}}\left\|\boldsymbol{D}_\tau(\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_{\tau-1}))\right\|\\ \leq&\frac{1}{\sqrt{\lambda_1}}\sum_{t=1}^{\tau}\left\|(\phi(\boldsymbol{b}_t\boldsymbol{Y})\otimes\boldsymbol{p}_t)(\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_{t-1}))\right\|\\ \leq&\frac{1}{\sqrt{\lambda_1}}\sum_{t=1}^{\tau}\left\|\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_{t-1})\right\|\\ \leq&\frac{1}{\sqrt{\lambda_1}}\sum_{t=1}^{\tau}\left\|\text{vec}(\boldsymbol{U}_*)-\text{vec}(\boldsymbol{U}_0)\right\|\times q^{t-1}\\ \leq&\frac{2\beta_u}{\sqrt{\lambda_1}}\times\frac{1-q^\tau}{1-q}\\ \leq&\frac{2\beta_u}{\sqrt{\lambda_1}(1-q)}\end{aligned}$$

The second inequality holds because Algorithm 1 updates parameters *online*. The fourth inequality is based on Uschmajew's work (Uschmajew, 2012; Wang et al., 2017), that the estimation of $\boldsymbol{U}$ is local $q$-linearly convergent to the optimizer.

Specifically, in the above inequations, parameter $q$ satisfies $0 < q < 1$. For conciseness, we denote $(1-q) \in (0,1)$ as $q \in (0,1)$ in Lemma 3.4. Some works (Liu et al., 2018) simply assume $\left\|\boldsymbol{A}_\tau^\top \boldsymbol{D}_\tau(\mathrm{vec}(\boldsymbol{U}_*) - \mathrm{vec}(\boldsymbol{U}_{\tau-1}))\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} = 0$ considering $\boldsymbol{U}_{t-1} \to \boldsymbol{U}_*$ when $t \to \infty$.

For the second term, we leverage the properties of *self-normalized vector-valued martingales* (Yadkori et al., 2011). Assuming $\epsilon_\tau$ belongs to a 1-sub-Gaussian process, then with probability at least $1 - \sqrt{\delta}$, there is the following inequality:

$$\left\|\boldsymbol{A}_\tau^\top \boldsymbol{\epsilon}_\tau\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} \leq \sqrt{\log \frac{\det(\boldsymbol{\mathcal{A}}_\tau)}{\det(\lambda_1 \boldsymbol{I}_{h^2})} - \log\delta}$$

Because of the Determinant-trace inequality, we have:

$$\begin{aligned}
&\det(\bar{\boldsymbol{\mathcal{A}}}_\tau) \leq \left(\frac{\mathrm{trace}(\bar{\boldsymbol{\mathcal{A}}}_\tau)}{h_*}\right)^{h_*} \leq \left(\lambda_1 + \frac{\tau}{h_*}\right)^{h_*} \Rightarrow \\
&\det(\boldsymbol{\mathcal{A}}_\tau) \approx \det(\bar{\boldsymbol{\mathcal{A}}}_\tau) \times \lambda_1^{h^2 - h_*} \leq \left(\frac{\mathrm{trace}(\bar{\boldsymbol{\mathcal{A}}}_\tau)}{h_*}\right)^{h_*} \times \lambda_1^{h^2 - h_*} \leq \left(\lambda_1 + \frac{\tau}{h_*}\right)^{h_*} \times \lambda_1^{h^2 - h_*}
\end{aligned}$$

In the above inequalities, $\bar{\boldsymbol{\mathcal{A}}}_\tau$ denotes the diagonal matrix whose diagonal entries are the eigenvalues of $\boldsymbol{\mathcal{A}}_\tau$ corresponding to the *effective dimensions*. It is worth noting that there may be a *small* coefficient on the right side of $\approx$, depending on the definition of the *effective dimension* $h_*$. However, we omit this coefficient for the sake of conciseness. Consequently, the second term has the following bound:

$$\left\|\boldsymbol{A}_\tau^\top \boldsymbol{\epsilon}_\tau\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} \leq \sqrt{\log \frac{\det(\boldsymbol{\mathcal{A}}_\tau)}{\lambda_1^{h_*}} - \log\delta} \leq \sqrt{h_* \log(1 + \frac{\tau}{\lambda_1 h_*}) - \log\delta}$$

According to the assumptions in Lemma 3.4, we have:

$$\lambda_1 \left\|\boldsymbol{\Theta}_*\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} \leq \lambda_1 \left\|\boldsymbol{\Theta}_*\right\| \leq \lambda_1 \beta_\theta$$

By integrating the above three terms, we complete the proof of the bound for $\alpha_\theta$ in Lemma 3.4. The proof for the bound of $\alpha_u$ follows a similar approach and is therefore omitted.

□

### A.4. Proof of Theorem 3.5

In this subsection, for description conciseness, the subscripts of functions are omitted when there is no risk of confusion. For example, we denote $r_{\tau+1}(\boldsymbol{b}_{\tau+1})$ simply as $r(\boldsymbol{b}_{\tau+1})$. Besides, we denote the acceptance estimated by $\bar{r}_{\tau+1}(\cdot) + e_{\tau+1}(\cdot)$ as $s(\cdot)$, and the samples as $\boldsymbol{\mu}_{\tau+1} = \phi(\boldsymbol{b}_{\tau+1}\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_{\tau+1})$ and $\boldsymbol{v}_{\tau+1} = \phi(\boldsymbol{b}_{\tau+1}\boldsymbol{Y}) \otimes \boldsymbol{p}_{\tau+1}$. Additionally, the optimal bid at time step $\tau+1$ is denoted as $\boldsymbol{b}_{\tau+1}^*$, thus $r(\boldsymbol{b}_{\tau+1}^*)$ and $r(\boldsymbol{b}_{\tau+1})$ are the true acceptance of the optimal bid $\boldsymbol{b}_{\tau+1}^*$ and the chosen bid $\boldsymbol{b}_{\tau+1}$ at $\tau+1$, respectively.

*Proof.* Firstly, we analyze Equation 14.

$$\boldsymbol{\mathcal{A}}_\tau(\phi(\boldsymbol{b}_{\tau+1}\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_{\tau+1}))^\top = (\boldsymbol{A}_\tau^\top \boldsymbol{A}_\tau + \lambda_1 \boldsymbol{I}_{h^2})\boldsymbol{\mu}_{\tau+1}^\top = \boldsymbol{A}_\tau^\top \bar{\boldsymbol{k}}_{\tau+1}^\top + \lambda_1 \boldsymbol{\mu}_{\tau+1}^\top$$

Rearranging the above equation, there is:

$$\begin{aligned}
\boldsymbol{\mu}_{\tau+1}^\top =& \boldsymbol{\mathcal{A}}_\tau^{-1}(\boldsymbol{A}_\tau^\top \bar{\boldsymbol{k}}_{\tau+1}^\top + \lambda_1 \boldsymbol{\mu}_{\tau+1}^\top) \\
=& \boldsymbol{A}_\tau^\top (\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1} \bar{\boldsymbol{k}}_{\tau+1}^\top + \lambda_1 \boldsymbol{\mathcal{A}}_\tau^{-1} \boldsymbol{\mu}_{\tau+1}^\top
\end{aligned}$$

The last equality above is based on the study of Haasdonk et al. (Haasdonk & Pekalska, 2010). As the kernel value at time step $\tau+1$ is denoted as $k_{\tau+1} = \boldsymbol{\mu}_{\tau+1}\boldsymbol{\mu}_{\tau+1}^{\mathsf{T}}$, there is:

$$
\begin{aligned}
\boldsymbol{\mu}_{\tau+1}\boldsymbol{\mu}_{\tau+1}^{\mathsf{T}} &= \bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}\bar{\boldsymbol{k}}_{\tau+1}^{\mathsf{T}} + \lambda_1 \boldsymbol{\mu}_{\tau+1}\boldsymbol{\mathcal{A}}_\tau^{-1}\boldsymbol{\mu}_{\tau+1}^{\mathsf{T}} \\
\Rightarrow \boldsymbol{\mu}_{\tau+1}\boldsymbol{\mathcal{A}}_\tau^{-1}\boldsymbol{\mu}_{\tau+1}^{\mathsf{T}} &= \frac{1}{\lambda_1}(k_{\tau+1} - \bar{\boldsymbol{k}}_{\tau+1}(\boldsymbol{K}_\tau + \lambda_1 \boldsymbol{I}_\tau)^{-1}\bar{\boldsymbol{k}}_{\tau+1}^{\mathsf{T}})
\end{aligned}
$$

Derivation for $\boldsymbol{v}_{\tau+1}\boldsymbol{\mathcal{D}}_\tau^{-1}\boldsymbol{v}_{\tau+1}^{\mathsf{T}}$ is similar. From the above results, Equation 14 is equivalent to the following format, consistent with existing UCB-based approaches. The remaining proof is based on this result.

$$
e_{\tau+1} = \alpha_\theta \sqrt{\boldsymbol{\mu}_{\tau+1}\boldsymbol{\mathcal{A}}_\tau^{-1}\boldsymbol{\mu}_{\tau+1}^{\mathsf{T}}} + \alpha_u \sqrt{\boldsymbol{v}_{\tau+1}\boldsymbol{\mathcal{D}}_\tau^{-1}\boldsymbol{v}_{\tau+1}^{\mathsf{T}}}
$$

Secondly, we prove that $s(\boldsymbol{b}^*_{\tau+1}) \geq r(\boldsymbol{b}^*_{\tau+1})$.

$$
\begin{aligned}
&s(\boldsymbol{b}^*_{\tau+1}) - r(\boldsymbol{b}^*_{\tau+1}) \\
=&\boldsymbol{\mu}^*_{\tau+1}(\mathrm{vec}(\boldsymbol{\Theta}_\tau) - \mathrm{vec}(\boldsymbol{\Theta}_*)) + \boldsymbol{v}^*_{\tau+1}(\mathrm{vec}(\boldsymbol{U}_\tau) - \mathrm{vec}(\boldsymbol{U}_*)) + \alpha_\theta \left\|\boldsymbol{\mu}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + \alpha_u \left\|\boldsymbol{v}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} \\
\geq& - \|\mathrm{vec}(\boldsymbol{\Theta}_\tau) - \mathrm{vec}(\boldsymbol{\Theta}_*)\|_{\boldsymbol{\mathcal{A}}_\tau} \left\|\boldsymbol{\mu}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} - \|\mathrm{vec}(\boldsymbol{U}_\tau) - \mathrm{vec}(\boldsymbol{U}_*)\|_{\boldsymbol{\mathcal{D}}_\tau} \left\|\boldsymbol{v}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} \\
&+ \alpha_\theta \left\|\boldsymbol{\mu}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + \alpha_u \left\|\boldsymbol{v}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} \\
\geq& - \alpha_\theta \left\|\boldsymbol{\mu}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} - \alpha_u \left\|\boldsymbol{v}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} + \alpha_\theta \left\|\boldsymbol{\mu}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + \alpha_u \left\|\boldsymbol{v}^*_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} = 0
\end{aligned}
$$

Thirdly, we bound $r(\boldsymbol{b}^*_{\tau+1}) \times f(\boldsymbol{b}^*_{\tau+1}) - r(\boldsymbol{b}_{\tau+1}) \times f(\boldsymbol{b}_{\tau+1})$. As $\boldsymbol{b}_{\tau+1}$ is the bid chosen by the NegUCB algorithm at time step $\tau+1$, we have:

$$
\begin{aligned}
&r(\boldsymbol{b}^*_{\tau+1}) \times f(\boldsymbol{b}^*_{\tau+1}) \leq s(\boldsymbol{b}^*_{\tau+1}) \times f(\boldsymbol{b}^*_{\tau+1}) \leq s(\boldsymbol{b}_{\tau+1}) \times f(\boldsymbol{b}_{\tau+1}) \\
\Rightarrow& r(\boldsymbol{b}^*_{\tau+1}) \times f(\boldsymbol{b}^*_{\tau+1}) - r(\boldsymbol{b}_{\tau+1}) \times f(\boldsymbol{b}_{\tau+1}) \\
\leq& \left\{\boldsymbol{\mu}_{\tau+1}\mathrm{vec}(\boldsymbol{\Theta}_\tau) + \boldsymbol{v}_{\tau+1}\mathrm{vec}(\boldsymbol{U}_\tau) + \alpha_\theta \left\|\boldsymbol{\mu}_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + \alpha_u \left\|\boldsymbol{v}_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} - \boldsymbol{\mu}_{\tau+1}\mathrm{vec}(\boldsymbol{\Theta}_*) - \boldsymbol{v}_{\tau+1}\mathrm{vec}(\boldsymbol{U}_*)\right\} \times f(\boldsymbol{b}_{\tau+1}) \\
=& \left\{\boldsymbol{\mu}_{\tau+1}(\mathrm{vec}(\boldsymbol{\Theta}_\tau) - \mathrm{vec}(\boldsymbol{\Theta}_*)) + \boldsymbol{v}_{\tau+1}(\mathrm{vec}(\boldsymbol{U}_\tau) - \mathrm{vec}(\boldsymbol{U}_*)) + \alpha_\theta \left\|\boldsymbol{\mu}_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + \alpha_u \left\|\boldsymbol{v}_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}}\right\} \times f(\boldsymbol{b}_{\tau+1}) \\
\leq& \left\{2\alpha_\theta \left\|\boldsymbol{\mu}_{\tau+1}\right\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + 2\alpha_u \left\|\boldsymbol{v}_{\tau+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}}\right\} \times f(\boldsymbol{b}_{\tau+1})
\end{aligned}
$$

The first inequality above is from the conclusion of the second proof step. Lastly, we prove the bound of cumulative regret. For the benefit function $f_t$ at time step $t$, we assume an union bound $\alpha_f$ such that $|f_t| \leq \alpha_f$ for $\forall \boldsymbol{b} \in B_t$ and $\forall t \in \{1, 2, ..., \tau\}$.

$$
\begin{aligned}
&\sum_{t=0}^{\tau} r(\boldsymbol{b}^*_{t+1}) \times f(\boldsymbol{b}^*_{\tau+1}) - r(\boldsymbol{b}_{t+1}) \times f(\boldsymbol{b}_{\tau+1}) \\
\leq& 2\alpha_\theta\alpha_f \sum_{t=0}^{\tau} \|\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_{t+1})\|_{\boldsymbol{\mathcal{A}}_\tau^{-1}} + 2\alpha_u\alpha_f \sum_{t=0}^{\tau} \left\|\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y}) \otimes \boldsymbol{p}_{t+1}\right\|_{\boldsymbol{\mathcal{D}}_\tau^{-1}} \\
\leq& 2\alpha_\theta\alpha_f \sqrt{\tau \sum_{t=0}^{\tau} \|\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y}) \otimes \phi(\boldsymbol{x}_{t+1})\|^2_{\boldsymbol{\mathcal{A}}_t^{-1}}} + 2\alpha_u\alpha_f \sqrt{\tau \sum_{t=0}^{\tau} \left\|\phi(\boldsymbol{b}_{t+1}\boldsymbol{Y}) \otimes \boldsymbol{p}_{t+1}\right\|^2_{\boldsymbol{\mathcal{D}}_t^{-1}}} \\
\leq& 2\alpha_\theta\alpha_f \sqrt{2\tau \log\frac{\det(\boldsymbol{\mathcal{A}}_\tau)}{\det(\lambda_1 \boldsymbol{I}_{h^2})}} + 2\alpha_u\alpha_f \sqrt{2\tau \log\frac{\det(\boldsymbol{\mathcal{D}}_\tau)}{\det(\lambda_2 \boldsymbol{I}_{mh})}} \\
\leq& 2\alpha_\theta\alpha_f \sqrt{2\tau h_* \log(1 + \frac{\tau}{\lambda_1 h_*})} + 2\alpha_u\alpha_f \sqrt{2\tau m_* \log(1 + \frac{\tau}{\lambda_2 m_*})}
\end{aligned}
$$

The third inequality is based on Lemma 11 of Abbasi-Yadkori's study (Yadkori et al., 2011), and the last inequality is based on Determinant-trace inequality to both $\bar{\boldsymbol{\mathcal{A}}}_\tau$ and $\bar{\boldsymbol{\mathcal{D}}}_\tau$. We do not explicitly emphasize the valid bid set $B_t$ in the above proof. However, the proof process remains the same if incorporating $B_t$.

□

### A.4.1. Cumulative Regret Analysis

The cumulative regret upper bound of NegUCB remains **independent of the cardinality of bids**, a notable distinction from existing algorithms like C2UCB (Qin et al., 2014), ComLinUCB (Wen et al., 2015), CC-MAB (Chen et al., 2018), etc., which exhibit upper bounds that are sub-linear concerning the cardinality of super arms. This behavior is attributed to the full-bandit feedback of negotiation problems and the Assumption 3.1. For instance, ComLinUCB estimates the reward of each arm, from which the general rewards of super arms are calculated, leading to error propagation. In contrast, NegUCB directly estimates the general rewards of super arms, eliminating error propagation. Assumption 3.1 requires that item contexts are defined by their basic features. Neglecting this principle may result in inaccuracies when capturing bid contexts. However, similar assumptions are commonly used in various applications, such as *recommendation* and *crowdsourcing*. Addressing this limitation in future research is encouraged. For full-bandit feedback, existing works such as DART (Agarwal et al., 2021), ETCG (Nie et al., 2022), RGL (Fourati et al., 2023), etc., have been developed. However, their regret upper bounds are contingent on the cardinality of super arms, as they lack consideration of contexts.

## B. Experiment

### B.1. Baseline Selection

Algorithms designed for contextual combinatorial bandits include C2UCB (Qin et al., 2014), ComLinUCB (Wen et al., 2015), CC-MAB (Chen et al., 2018), CN-UCB (Hwang et al., 2023), and others. However, these algorithms are tailored for semi-bandit feedback and cannot be directly applied to our specific problems. Our experiments adapt LinUCB (Li et al., 2010) to combinatorial bandits with full-bandit feedback. Specifically, we extend LinUCB by incorporating Assumption 3.1, which treats bids as the basic arms of the original algorithm. The algorithms DART (Agarwal et al., 2021), ETCG (Nie et al., 2022), and RGL (Fourati et al., 2023) are specifically tailored for full-bandit feedback. However, they do not consider contextual information, rendering them unsuitable for addressing our particular problems.

Neural network-based algorithms, e.g., Neural-UCB, CN-UCB, or variants, are not chosen as the baselines in our experiments. Although neural networks have powerful representation capabilities, the networks in neural bandits cannot be large, as the computational complexity is cubic concerning the number of network parameters, limiting their capabilities. Neural-LinUCB (Xu et al., 2022) solely explores the output layer to expedite the neural-bandit algorithms. Nevertheless, it encounters instability issues in the iteration process among three components: learning the parameters $\boldsymbol{\Theta}$, $\boldsymbol{U}$, and learning the neural network $\phi$. According to exiting works (Liu et al., 2018), prior knowledge or additional constraints are required to govern learning under these scenarios.

*Table 2.* Exploration parameter settings for each algorithm.

| Algorithm | | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ | $\alpha_6$ |
|---|---|---|---|---|---|---|---|
| LinUCB | | 0 | 1 | 4 | **8** | 16 | 32 |
| ANAC agent | | 0 | **1** | 2 | 3 | 4 | 5 |
| FactorUCB | # items is 5 | 0 | 0.4 | **0.8** | 1.2 | 1.6 | 2 |
| | # items is 20 | 0 | **0.1** | 0.4 | 0.6 | 0.8 | 1 |
| KernelUCB | | 0 | **1** | 2 | 4 | 6 | 8 |
| NegUCB | | 0 | **0.1** | 0.4 | 0.6 | 0.8 | 1 |

In addition to bandit-based algorithms, alternative methods for negotiation have been proposed (Cao et al., 2018; Buron et al., 2019; Bagga et al., 2020). However, as discussed earlier, these approaches often struggle to address the exploitation-exploration dilemma and handle large action spaces effectively. Consequently, they are anticipated to exhibit inferior performance compared to NegUCB. In our experiments, we adopt a variant of ANEGMA (Bagga et al., 2020). The notable difference is the absence of the pre-training component, caused by the unavailability of historical negotiation data for various negotiation tasks. To compensate for this omission, we extend the training duration of ANEGMA, ensuring that the results accurately reflect its true capabilities.

## B.2. Bid Design

In this subsection, we illustrate the design of bid vectors for each task through illustrative examples. These bids are structured based on the NegUCB algorithm and tailored to specific negotiation problems. However, it's advisable to flexibly adjust bid formats to accommodate diverse problems and algorithms.

### B.2.1. Multi-issue Negotiation

Consider a negotiation scenario with four issues, each having $4, 2, 2, 3$ possible values, respectively. The bid vector, representing the potential outcomes for these issues, is of size $4 + 2 + 2 + 3 = 11$. For instance, a bid expressing *value 3* for issue A, *value 1* for issue B, *value 2* for issue C, and *value 2* for issue D is denoted by the bid vector $\boldsymbol{b} = (0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0)$. It signifies the selection of values for each issue.

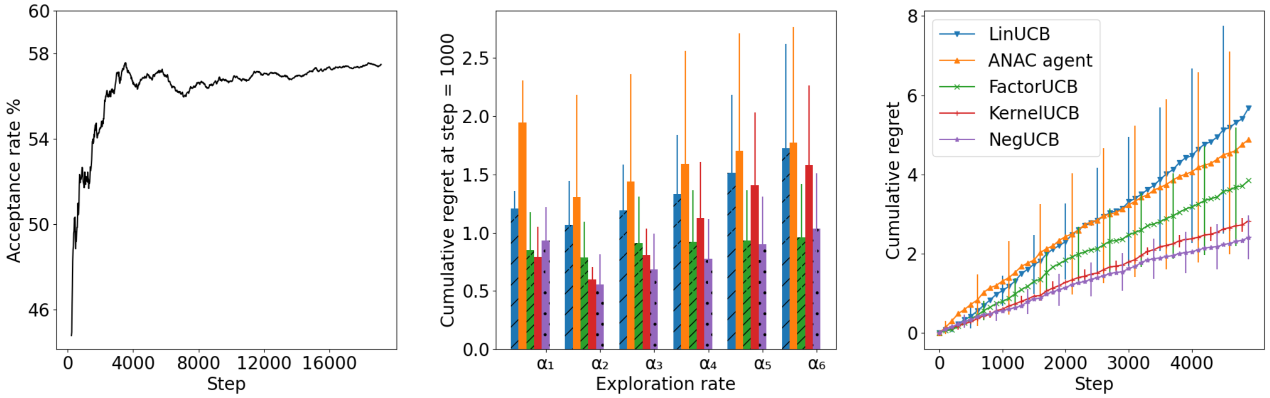
### B.2.2. Resource Allocation

Figure 2 provides a simple example of how a bid can be designed in a resource allocation task, where each negotiator takes distinct categories of items. However, both negotiators may want some items in the same category. For example, both of them want some apples. Consequently, the bids for this task in our experiment are designed in a more general way. Let's assume three types of items are available: 4 strawberries, 2 peppers, and 5 apples. A bid representing a request for 1 strawberry, 1 pepper, and 2 apples, while the counterpart negotiator retains the remaining 3 strawberries, 1 pepper, and 3 apples, can be denoted by the bid vector $\boldsymbol{b} = (1, 1, 2, -3, -1, -3)$. The first three entries represent the items requested by our negotiator, while the remaining three represent those for the counterpart. Under this setting, the item context matrix is:

$$\boldsymbol{Y} = \begin{pmatrix} 2 & 1 \\ 1 & 3 \\ 5 & 4 \\ - - - & - - - \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix} \tag{19}$$

### B.2.3. Trading

For a trading scenario involving three types of items—strawberries, peppers, and apples—a bid indicating that our negotiator offers 1 pepper and 1 strawberry to the counterpart while seeking 2 apples in return can be represented by the bid vector $\boldsymbol{b} = (1, 1, 0, 0, 0, -2)$. The first three entries signify the items provided by our negotiator, whereas the remaining three entries denote the items sought from the counterpart. The item context matrix is similar to that in Equation 19.



(a) Acceptance by reinforcement learning (b) × 100: Cumulative acceptance regret (c) ×100: Cumulative acceptance regret

*Figure 8.* More experiment results on resource allocation task.

### B.3. More Experiment Results and Analysis to *Resource Allocation*

Because the exploitation scales of algorithms vary, their exploration scales also vary largely. We conduct a search and summarize the six representative exploration rates corresponding to Figure 4(a) in Table 2, where the rates in bold are the optimal ones of each algorithm. Figure 8(a) presents the outcomes of the reinforcement learning method. Due to the random nature of exploration in reinforcement learning, employing $\epsilon$-greedy exploration, we extend the training duration to 20000 steps to ensure the results accurately depict the algorithm's true capabilities. As depicted in Figure 8(a), the acceptance rate exhibits an initial increase but soon plateaus, facing challenges in surpassing an acceptance rate of 0.6.

In Figure 5, the acceptance rates exhibit initial fluctuations due to the limited negotiation steps, resulting in sharp increases when deals occur, primarily caused by randomness. It is important to note that these initial spikes do not necessarily imply high negotiation capabilities. Other studies have documented similar observations, such as TCB (Liu et al., 2018) and FactorUCB (Wang et al., 2017).

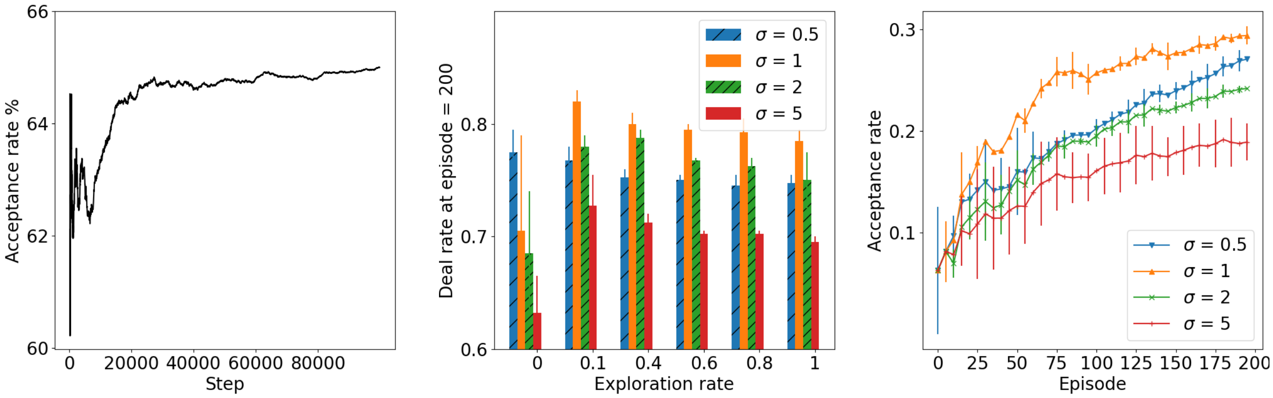
Additionally, we add one more experiment with a larger action space. Assuming there are three categories of items and the number of items in each category does not exceed 20, the action space comprises at most $21 \times 21 \times 21 = 9261$ actions. Experiment results under this setting are shown in Figure 8(b), Figure 8(c), and Figure 9(a), demonstrating the advantages of NegUCB compared to the baselines. It is worth noting that the complexity of the task is not solely determined by the size of the action space but also by other variables, such as the item contexts and the attitudes of the counterparts, which may be the reason why the acceptance rates of the added experiment are higher than those in the main content.

### B.4. More Experiment Results and Analysis to *Trading*

In the CivRealm experiment, we have chosen a bid cardinality of $\gamma = 4$ for the sake of experiment efficiency. Setting $\gamma$ too large would require additional search algorithms to find the optimal bid in Equation 1, which falls beyond the scope of this work and introduces errors unrelated to our algorithm. Moreover, bids in *trading* often consist of only a few items. However, this is not contradictory to the large action space issue, as the action space has a cardinality of $\sum_{j=1}^{\gamma} \binom{n}{j}$, which can still be large even for a small $\gamma$.

The contexts of negotiator pairs encompass the technologies our negotiator and the counterpart possess. Contexts of technologies include metrics such as *cost*, *research_reqs_count*, and *num_reqs*, which are provided by CivRealm (Qi et al., 2024) and describe fundamental features of technologies.

In this experiment, we employ the SE kernel given by $\kappa(\boldsymbol{x}_w, \boldsymbol{x}_j) = \exp(-\frac{1}{2\sigma^2} \left\| \boldsymbol{x}_w - \boldsymbol{x}_j \right\|^2)$. We systematically explore SE kernels with diverse hyper-parameters $\sigma$, specifically $\sigma = 0.5, 1, 2,$ and $5$, to fine-tune the most suitable kernel function for the trading task discussed in this subsection. The results, illustrated in Figure 9(b), display the final deal rates after 200 episodes using different kernel functions. According to the experiment results from CivRealm (Qi et al., 2024), their deal rates are less than 0.4, notably inferior to those of NegUCB. Based on Figure 9(b), we conclude that the SE kernel with $\sigma = 1$ emerges as the most suitable choice for the trading task on CivRealm. Additionally, Figure 9(c) further illustrates



(a) Acceptance by reinforcement learning (b) Deal rate w.r.t. exploration rate (c) Acceptance rate

*Figure 9.* More experiment results on resource allocation and trading tasks. *Deal rate* is a metric defined by CivRealm, quantifying the percentage of episodes that result in a deal, while the *acceptance rate* is the percentage of the proposed bids being accepted..

the acceptance rates of NegUCB with various SE kernels under their corresponding optimal exploration rates, specifically $\alpha_\theta = \alpha_u = 0, 0.1, 0.4, 0.1$ for $\sigma = 0.5, 1, 2, 5$, respectively.