Combinatorial Partial Monitoring Game with Linear Feedback and Its Applications

Tian Lin[*] LINT10@MAILS.TSINGHUA.EDU.CN
Bruno Abrahao[†] ABRAHAO@CS.CORNELL.EDU
Robert Kleinberg[†] RDK@CS.CORNELL.EDU
John C.S. Lui[‡] CSLUI@CSE.CUHK.EDU.HK
Wei Chen[§] WEIC@MICROSOFT.COM
[*]Tsinghua University, Beijing, China
[†]Cornell University, Ithaca, NY 14850, USA
[‡]The Chinese University of Hong Kong, Shatin, NT, Hong Kong
[§]Microsoft Research, Beijing, China

Abstract

In online learning, a player chooses actions to play and receives reward and feedback from the environment with the goal of maximizing her reward over time. In this paper, we propose the model of combinatorial partial monitoring games with linear feedback, a model which simultaneously addresses limited feedback, infinite outcome space of the environment and exponentially large action space of the player. We present the Global Confidence Bound (GCB) algorithm, which integrates ideas from both combinatorial multi-armed bandits and finite partial monitoring games to handle all the above issues. GCB only requires feedback on a small set of actions and achieves $O(T^{\frac{2}{3}} \log T)$ distribution-independent regret and $O(\log T)$ distribution-dependent regret (the latter assuming unique optimal action), where T is the total time steps played. Moreover, the regret bounds only depend linearly on $\log |\mathcal{X}|$ rather than $|\mathcal{X}|$, where $\mathcal{X}$ is the action space. GCB isolates offline optimization tasks from online learning and avoids explicit enumeration of all actions in the online learning part. We demonstrate that our model and algorithm can be applied to a crowdsourcing application leading to both an efficient learning algorithm and low regret, and argue that they can be applied to a wide range of combinatorial applications constrained with limited feedback.

1. Introduction

In the online learning framework, a player (or a learner) and the environment interact with each other in discrete time steps in the following way. At each time step, the environment generates an outcome and the player selects an action, and when the action is applied to the environment outcome, the player gains a reward and receives a feedback about the environment outcome and potentially about her reward. The player wants to learn from past feedbacks to maximize her reward over time, or as in standard treatment, minimize her *regret*, which is the difference between the reward she could collect if she always plays the best action over time and the reward she actually receives. We consider stochastic online learning in this paper, which means that the environment generates outcomes following a certain predetermined distribution not known to the player. The key in designing the player's learning algorithm is to address the tradeoff between exploration and exploitation: the player needs to try all possible actions (exploration) while biasing toward the actions giving her the best rewards so far (exploitation).

Online learning has a wide range of applications and has been extensively studied over the years in statistics and machine learning. The multi-armed bandit (MAB) problem (Robbins, 1985) is a classical framework in online learning in which the environment outcome in each time step is the reward of n *arms* (or actions), the player's action is to select one of the n arms to play, and the reward the player receives, as well as her feedback, is the reward of the selected arm. Algorithms for MAB achieve $O(\log T)$ distribution-dependent regret (i.e. the regret also depends on the distribution of outcomes) and $O(\sqrt{T})$ distribution-independent regret, over T time steps (Auer et al., 2002).

In some applications with combinatorial structures, the number of arms to play may be exponential in the problem

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

instance size, and MAB becomes impractical since both its regret and its running time would be exponential in the problem size. The combinatorial MAB (CMAB) problem (Chen et al., 2013) was proposed to address this issue. In CMAB, each action is a *super arm*, which is a set of base arms, and once played, the outcomes of all base arms are revealed to the player as the feedback. CMAB algorithms can achieve the same level of regret as MAB algorithms in terms of dependence on the time horizon T, and both the running time and the regret of CMAB algorithms is polynomial in the problem size when assuming the existence of a polynomial-time offline optimization oracle.

MAB and CMAB require feedback on the reward or outcome of the chosen arm, or each of its constituent base arms in the case of CMAB. This feedback may not be available in many applications. Partial monitoring games are online learning models addressing the issue of limited feedback information (Piccolboni & Schindelhauer, 2001; Cesa-Bianchi et al., 2006; Antos et al., 2012). Existing works focus on a finite action space $\mathcal{X}$ and a finite outcome space, and allow flexible feedback structure for each action $x \in \mathcal{X}$. Depending on the condition governing feedback and reward, they could achieve either $\Theta(T^{\frac{2}{3}})$ or $\tilde{\Theta}(\sqrt{T})$ distribution-independent regret in T time steps.[1] However, existing work on partial monitoring does not address the case of exponential action space in that both the regret and the running time of the learning algorithm depend linearly or polynomially on the size of the action space $\mathcal{X}$. Moreover, they do not address the case of infinite outcome space.

Although CMAB and current partial monitoring work address limited feedback, exponential action space, and infinite outcome space separately, some applications may require them all simultaneously. Consider the following motivating example in crowdsourcing. Suppose that a crowdsourcing platform acts as a broker between workers who have time and skill and employers who have tasks to complete. Every day employers post tasks, and the crowdsourcing platform matches workers to tasks. The platform wants to maximize the social welfare over time, which is the total performance of matched worker-task pairs, and it relies on employers and workers to provide feedback on the performance of completed tasks. However, providing task performance feedback is inconvenient or costly for employers and workers, and in some cases may also violate privacy concerns. Therefore, the platform can only ask employers and workers to provide partial feedback for a small subset of completed tasks. Moreover, the action space in this case is the set of all matchings between workers and tasks, which is exponential in the problem size. Finally, the performance feedback may be measured as a continuous variable such as time to complete the task, payment for the task, etc., leading to an infinite outcome space. Thus this application requires a learning method handling limited feedback, exponential action space, and infinite outcome space at the same time, which has not been addressed in the literature to the best of our knowledge.

In this paper, we fill this gap by combining ideas from both CMAB and partial monitoring and providing a unified model and algorithm addressing all the above issues. We first extend the finite partial monitoring model with infinite and continuous outcomes and linear feedback as follows. We model the outcome of the environment as a vector of bounded and continuous random variables $\mathbf{v}$, thus supporting an infinite outcome space. For each action $x \in \mathcal{X}$, the feedback of x on $\mathbf{v}$ is given by the linear transformation $M_x\mathbf{v}$, where M_x is the transformation matrix of x. The matrices M_x provide a very flexible model of limited feedback, including the extreme that an all-zero matrix M_x means that there is no feedback for action x. We refer this model as a *combinatorial partial monitoring game with linear feedback*.

We present a general algorithm called Global Confidence Bound (GCB) for combinatorial partial monitoring games with linear feedback with distribution-independent regret bound $O(T^{\frac{2}{3}} \log T)$ and distribution-dependent bound $O(\log T)$ (the latter assuming unique optimal action). One of the key ideas of GCB is to use a small observer set of actions for exploration. This idea combines the observer set idea from partial monitoring with the idea of estimating the small base arm set from CMAB. Our algorithm handles the large action space in that (a) our regret bound linearly depends on $\log |\mathcal{X}|$ instead of $|\mathcal{X}|$, and (b) our algorithm does not enumerate actions in the online learning part but only relies on efficient problem-specific solutions to the corresponding offline optimization tasks, similar to the way large action spaces are handled in CMAB.

We then apply GCB to the crowdsourcing application and show that it allows an efficient polynomial-time learning algorithm with low regret depending linearly on $\log |\mathcal{X}|$, polynomially on problem instance size, besides being sublinear in the time horizon T. It is important to point out that our algorithm GCB not only applies to the crowdsourcing application but to a much larger class of applications. Many matching applications, whether they are online dating brokages matching individuals, recommender systems matching products with customers, or advertising systems matching advertisements with users, could potentially benefit from the algorithm since they may also have constraints limiting the feedbacks the systems may obtain. It also goes beyond matching tasks and may work well with other combinatorial optimization tasks that has efficient offline solutions, such as online shortest path learning and optimization. Furthermore, we formulate GCB in a general setting

[1] The notations $\tilde{O}, \tilde{\Theta}$ disregard polylogarithmic factors.

in which the reward function may not be linear in the outcome vector $\mathbf{v}$, and only a continuity assumption is needed on the expected reward function, which may further enlarge the application range of our algorithm.

To summarize, the contributions of our paper include: (a) we present a general online learning model and algorithm that handle limited feedback, exponential action space and infinite outcome space simultaneously and work with nonlinear reward functions; (b) we provide a detailed formulation and solution for partial monitoring games with linear feedback;[2] and (c) we propose an application in the domain of crowdsourcing worker allocation, and demonstrate that our algorithm can be applied to the application yielding an efficient algorithm with low regret.

The rest of the paper is organized as follows. Section 2 discussed related work. Section 3 provides the details of our model and assumptions. Section 4 presents the GCB algorithm and its regret bound. Section 5 describes the crowdsourcing application and demonstrates how GCB is applied to the application. We conclude the paper in Section 6. Detailed regret analysis are provided in the supplementary material.

2. Related Work

Feedback models in online learning (Cesa-Bianchi & Lugosi, 2006) have been studied for a long time. The simplest model of feedback is a *full-information* game. When one action is played, feedbacks directly include rewards of all actions. The regret for a full-information game is $\Theta(\sqrt{T \log |\mathcal{X}|})$ (Littlestone & Warmuth, 1989; Vovk, 1990). Another well-studied feedback model is a *multi-armed bandit* game, introduced by (Robbins, 1985). (The name is derived from an analogy of multiple slot machines.) In each time step one action is played, and the reward of that arm is revealed as feedback. The tight regret bound for adversarial multi-armed bandit games is $\Theta(\sqrt{|\mathcal{X}|T})$ (Audibert & Bubeck, 2009). Since only one reward is revealed while others are unknown, one faces an *exploration vs. exploitation* dilemma. The UCB1 algorithm for stochastic multi-armed bandit games (Auer et al., 2002) is a simultaneous exploration and exploitation policy, which applies a *confidence bound* on each arm, and achieves a distribution-dependent regret bound of $O(\log T)$ over time. The above literature distinguishes two kinds of games: adversarial games and stochastic games (Bubeck & Cesa-Bianchi, 2012), and we focus on the latter in this paper.

A recent line of research focuses on handling combinatorial structures in adversarial (Cesa-Bianchi & Lugosi, 2012) or stochastic online learning (Gai et al., 2012; Chen et al., 2013) and provides useful ideas which we incorporate here. In (Chen et al., 2013), the authors use an offline oracle to separate the problem-specific solver and the learning algorithm, enabling the possibility of dealing with large sets of actions.

Another line of research aims to characterize the difficulty of learning in different feedback models. Finite partial monitoring games were first defined in (Piccolboni & Schindelhauer, 2001). Studies in (Cesa-Bianchi et al., 2006; Antos et al., 2012) characterize finite partial monitoring games into four categories: *trivial*, *easy*, *hard*, and *hopeless*, with regret bounds of 0, $\tilde{\Theta}(T^{\frac{1}{2}}), \Theta(T^{\frac{2}{3}})$, and $\Theta(T)$ respectively. Algorithm BALATON in (Antos et al., 2012) achieved $\tilde{\Theta}(T^{\frac{1}{2}})$ assuming the local observability property to separate different optimal actions, and CBP in (Bartók et al., 2012) is proposed to switch between $\tilde{O}(T^{\frac{1}{2}})$ and $\tilde{O}(T^{\frac{2}{3}})$ adaptively, when the problem may be either locally or globally observable. In constrast, not only is our model able to handle exponentially large action space, but we are also the first to provide an $O(\log T)$ distribution-dependent regret for a globally observable game (with the unique optimal action assumption), whereas CBP only provides an $O(\log T)$ distribution-dependent regret for locally observable games, which is a stronger assumption.

3. Model and Definitions

In this paper, we consider the *combinatorial stochastic partial monitoring game with linear feedback*, as described below. A combinatorial partial monitoring game is a repeated game between the player and the environment at discrete time steps $t = 1, 2, \cdots$. Let $v(t)$ denote the value of a variable v at the end of time t. Before the game starts, the environment determines a fixed probability distribution p on $[0,1]^n$, where n is the dimension, and p is not revealed to the player. At each time t, the environment samples an independent random vector[3] $\mathbf{v}(t) \in [0,1]^n$ from the fixed distribution, and this vector is taken to be the environment's outcome at this time step. The random outcome vector $\mathbf{v}(t)$ is not observed by the player. Meanwhile, at time t the player chooses an action $x(t) \in \mathcal{X}$, where $\mathcal{X}$ is a finite set of all possible actions of the player. When the action $x(t)$ is applied to the environment outcome $\mathbf{v}(t)$, the player gains a reward[4] $r(x(t), \mathbf{v}(t)) \in \mathbb{R}$ and receives a feedback $\mathbf{y}(t)$, which is a linear transformation of the latent outcome $\mathbf{v}(t)$ as we now explain. For every action $x \in \mathcal{X}$, there is a

[2] The final paragraph in the conclusion of (Bartók et al., 2011) mentions a similar extension to linear feedback without further details. However, our algorithm and analysis further focus on handling exponential action space, which is not addressed in the framework of (Bartók et al., 2011).

[3] All vectors in the paper are column vectors.

[4] Some prior work treats loss rather than reward. Reward and loss are two symmetric terms, and our results apply to the case of loss functions too after performing the appropriate sign changes.

transformation matrix $M_x \in \mathbb{R}^{m_x \times n}$, whose row dimension m_x depends on x. At time t, the player receives the feedback vector $\mathbf{y}(t) = M_{x(t)} \cdot \mathbf{v}(t) \in \mathbb{R}^{m_{x(t)}}$, and this is what we mean by "linear feedback". Note that the actual reward $r(x(t), \mathbf{v}(t))$ may not be part of the feedback $\mathbf{y}(t)$ and thus may not be observed by the player. Also M_x for some action x could be all zeros, meaning that there is no feedback for action x. The outcome dimension n, the action space $\mathcal{X}$, the reward function $r(\cdot, \cdot)$ and transformation matrices $M_x, \forall x \in \mathcal{X}$ form the instance of the game, and are known to the player. For some game instances (e.g. the crowdsourcing application of Section 5), $\mathcal{X}$ and M_x's have certain combinatorial structure and thus they have a succinct representation, and the size of $\mathcal{X}$ may be exponential to the size of the (succinctly represented) game instance.

To summarize, the game proceeds at discrete time steps $t = 1, 2, \ldots$, and at each time t:

1. The environment samples an outcome vector $\mathbf{v}(t)$ from its predetermined distribution p independently.
2. The player chooses an action $x(t)$ from $\mathcal{X}$, which could be based on the feedback the player receives in the previous time steps.
3. When the action $x(t)$ is applied to the environment outcome $\mathbf{v}(t)$, the player gains the reward $r(x(t), \mathbf{v}(t))$, and receives the feedback $\mathbf{y}(t) = M_{x(t)} \cdot \mathbf{v}(t)$.

Let $\mathbf{v}$ be a generic random outcome vector sampled from the outcome distribution p. As is common, we use the expected regret metric to measure the performance of the player. Given a time horizon T, the *expected regret*, or simply *regret* at time T, is defined as:

$$R(T) = T \max_{x \in \mathcal{X}} \mathbb{E}\left[r(x, \mathbf{v})\right] - \sum_{t=1}^{T} \mathbb{E}\left[r(x(t), \mathbf{v}(t))\right]. \quad (1)$$

In the first term above, the expectation is taken from the distribution of $\mathbf{v}$, and it represents the reward the player would get if she always played the action that yields the maximum expected reward at each time step. In the second term, the expectation is taken from the distribution of $\mathbf{v}(t)$ and the possible randomness of the player's algorithm, and it represents the expected reward the player actually gains by time T when running her algorithm.

Let $\boldsymbol{\nu} = \mathbb{E}[\mathbf{v}]$ denote the vector of expected values of $\mathbf{v}$. From Equation (1), we see that what we are interested in is the expected reward of playing an action x, namely $\mathbb{E}[r(x, \mathbf{v})]$, where the expectation is taken over the distribution of $\mathbf{v}$. In this paper, we assume that the above expected reward is a function of x and the expected outcome vector $\boldsymbol{\nu}$ of $\mathbf{v}$, and thus we define $\overline{r}(x, \boldsymbol{\nu}) = \mathbb{E}[r(x, \mathbf{v})]$. This assumption is always satisfied if the reward function $r(x, \mathbf{v})$ is a linear function of $\mathbf{v}$, or if the full distribution of $\mathbf{v}$ is determined by its mean vector $\boldsymbol{\nu}$, as in the case of a vector of independent Bernoulli random variables. Let x^* be the optimal action for the expected reward under the expected outcome vector $\boldsymbol{\nu}$, i.e. $x^* = \max_{x \in \mathcal{X}} \overline{r}(x, \boldsymbol{\nu})$. Then we can rewrite the regret in Equation (1) as follows:

$$R(T) = T \cdot \overline{r}(x^*, \boldsymbol{\nu}) - \sum_{t=1}^{T} \mathbb{E}\left[\overline{r}(x(t), \boldsymbol{\nu})\right], \quad (2)$$

where the expectation in the second term is taken over the randomness of $x(t)$, which could come from either the random feedback the player received in the previous time steps or the randomness in the player's algorithm itself.

3.1. Assumptions

In this section, we provide and justify two technical assumptions needed for our proposed algorithm to work.

After action $x(t)$ is played at time t, the feedback $\mathbf{y}(t)$ is a linear transformation of the outcome $\mathbf{v}(t)$, and thus the information about the environment outcome is distorted. To retrieve an estimate of $\mathbf{v}(t)$, we assume the existence of a global observer set, which is defined below following a similar definition in (Piccolboni & Schindelhauer, 2001).

Definition 3.1 (Global Observer Set)**.** *For a set of actions $\sigma = \{x_1, x_2, \cdots, x_{|\sigma|}\} \subset \mathcal{X}$, we stack*[5] *their transformation matrices to obtain a $(\sum_{i=1}^{|\sigma|} m_{x_i})$-by-$n$ block matrix $M_\sigma = \left(M_{x_1}; M_{x_2}; \cdots; M_{x_{|\sigma|}}\right)$. We say that σ is a* global observer set *if M_σ is of full column rank, i.e. $rank(M_\sigma) = n$. This implies that the Moore-Penrose pseudoinverse M_σ^+ satisfies $M_\sigma^+ M_\sigma = I_n$, where I_n is an n-by-n identity matrix.*

Our first assumption is thus:

- **Existence of global observer set.** We assume that there exists a global observer set σ in the partial monitoring game with the linear feedback transformation matrices M_x for all $x \in \mathcal{X}$.

The above assumption is reasonable, since if it does not hold, it means that even if we are allowed to play all possible actions in $\mathcal{X}$ on the same given outcome $\mathbf{v}$, we cannot recover the value of $\mathbf{v}$. This would mean that we may not be able to detect the difference between the rewards on different actions and may not achieve sublinear regret. This is similar to the case of the hopeless problem class with regret no better than $\Theta(T)$ (Cesa-Bianchi et al., 2006).

When the above assumption holds, one can systematically find a global observer set σ with $|\sigma| \leq n$. For example, we can add an action x into σ if it strictly increases the

[5] Notation $(A; B; C)$ means stacking up matrix A on top of B and then on top of C, where A, B, and C have same column dimensions.

rank of the resulting M_σ, until M_σ reaches the full rank n. As a consequence, the size of the global observer set is small. Note that the above assumption implies that we do not require feedback for any actions outside the global observer set σ.

Our second assumption is on the continuity of the expected reward function.

- **Continuity of the expected reward function.** We assume the expected reward function is Lipschitz continuous in its second variable, with Lipschitz constant $L > 0$; that is, for all $x \in \mathcal{X}$, for any two mean outcome vectors $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \in [0,1]^n$, we have $|\overline{r}(x, \boldsymbol{\nu}_1) - \overline{r}(x, \boldsymbol{\nu}_2)| \le L\|\boldsymbol{\nu}_1 - \boldsymbol{\nu}_2\|_2$. [6]

The above continuity assumption is also a reasonable assumption, since without such an assumption, a small change in the mean outcome vector may result in an arbitrarily large change in the expected reward, which means that it is very difficult to control the regret bound when one has to learn the mean outcome vector over time. The assumption certainly holds when the reward function is a linear function of the outcome vector.

The above continuity assumption, together with the bounded support of the distribution of $\mathbf{v}$, also implies that the expect reward in a given game instance is bounded. In fact, for each $x \in \mathcal{X}$ and each $\boldsymbol{\nu} \in [0,1]^n$, $|\overline{r}(x,\boldsymbol{\nu})| \le |\overline{r}(x,\mathbf{0})| + |\overline{r}(x,\boldsymbol{\nu}) - \overline{r}(x,\mathbf{0})| \le |\overline{r}(x,\mathbf{0})| + L\|\boldsymbol{\nu} - \mathbf{0}\|_2 \le |\overline{r}(x,\mathbf{0})| + L\sqrt{n}$. Thus, $|\overline{r}(x,\boldsymbol{\nu})| \le \max_{x\in\mathcal{X}} |\overline{r}(x,\mathbf{0})| + L\sqrt{n}$. Let $R_{\max} = \max_{x_1,x_2\in\mathcal{X},\boldsymbol{\nu}\in[0,1]^n} |\overline{r}(x_1,\boldsymbol{\nu}) - \overline{r}(x_2,\boldsymbol{\nu})|$ be the maximum difference in expected reward among actions for any possible outcome distribution. We will use $R_{\max}$ in our regret bound later.

3.2. Relationship with finite partial monitoring game

It is easy to see that our model encompass MAB, full-information game, and CMAB as special cases, while our model focuses on limited feedback not addressed in these other models, as already discussed in the introduction. Thus, we only provide some further comparison with the finite partial monitoring game model.

The finite stochastic partial monitoring game (Bartók et al., 2011) generalizes the MAB and the full-information game and focuses on partial feedback information. It consists of N actions and M possible outcomes. At each time t the player selects one action $I(t)$ and the environment samples one outcome $J(t)$ from a predetermined distribution, and the player incurs a loss $L(I(t), J(t))$ and receives a feedback $H(I(t), J(t)) \in \Sigma$, where Σ is a finite symbol set, L is called the *loss matrix* and H is called the *feedback matrix*. In our setting, we could model the outcome vector as $\mathbf{v} \in \{e_i \,|\, i = 1, 2, \ldots, M\}$, where e_i is the M-dimensional 0-1 vector with only the i-th entry being 1. The transformation matrix M_x is the signal matrix of action x defined in (Bartók et al., 2011). The reward function $r(x, \mathbf{v}) = -L(x, \cdot)\mathbf{v}$, where $L(x, \cdot)$ is the row of L corresponding to action x. Since the reward function is linear, the continuity of the expected reward function holds. Our assumption on the existence of the global observer set is slightly stronger than the global observable property in (Piccolboni & Schindelhauer, 2001) to accommodate the more general class of nonlinear reward functions. Our model is more general than the finite stochastic partial monitoring game, in that it allows infinite outcome spaces, general linear transformation from outcomes to observations, and possibly non-linear reward functions.

4. Learning Algorithm GCB

Our learning algorithm integrates ideas from both CMAB and partial monitoring. In CMAB, in order to handle exponentially large action space, the algorithm utilizes a small set of base arms and their outcome estimates to derive the rewards of all actions. In partial monitoring, in order to handle limited feedback, it uses observer sets so that feedback from actions in an observer set helps in estimating rewards of other actions. In our combinatorial partial monitoring model, we combine the above ideas and use the global observer set σ as defined in Section 3. The global observer set both helps to estimate rewards of other non-observable actions as in partial monitoring, and handles exponentially large action space because it is small.

More specifically, given an observer set $\sigma = \{x_1, x_2, \cdots, x_{|\sigma|}\}$, when the player plays these actions at time steps $t_1, t_2, \cdots, t_{|\sigma|}$, the environment produces sample outcomes $\mathbf{v}(t_1), \mathbf{v}(t_2), \cdots, \mathbf{v}(t_{|\sigma|})$, and the feedbacks can be stacked as $\big(\mathbf{y}(t_1); \mathbf{y}(t_2); \cdots; \mathbf{y}(t_{|\sigma|})\big) = \vec{\mathbf{y}}(t)$ at final step $t = t_{|\sigma|}$. From the definition of σ, we can denote the inverse function as $\mathbf{I}(M_\sigma, \vec{\mathbf{y}}) = M_\sigma^+ \vec{\mathbf{y}}$. (Recall that M_σ^+ denotes the Moore-Penrose pseudoinverse of the stacked matrix M_σ.) After the inversion, the estimate $\tilde{\mathbf{v}}(t)$ of outcomes is obtained at time t: $\tilde{\mathbf{v}}(t) = \mathbf{I}(M_\sigma, \vec{\mathbf{y}}(t))$. It is easy to check that $\mathbb{E}\left[\tilde{\mathbf{v}}(t)\right] = \boldsymbol{\nu}$, which implies that we can use multiple independent values of $\tilde{\mathbf{v}}(t)$ to obtain an accurate estimate of $\boldsymbol{\nu}$. We call the above one *round of estimation* or one *round of exploration* of the player. Our algorithm invokes multiple rounds of exploration to obtain an accurate estimate of $\boldsymbol{\nu}$, which in turn helps us to estimate the rewards of all actions.

Denote noise $\varepsilon(t) = \mathbf{v}(t) - \boldsymbol{\nu}$, thus $\|\varepsilon(t)\|_2 \le \sqrt{n}$. Since $M_\sigma^+ M_\sigma = I_n$ and $M_\sigma^+ = (M_\sigma^\intercal M_\sigma)^{-1} M_\sigma^\intercal$, the estimated

[6] For technical reasons, define $\phi(\boldsymbol{\nu}) = \max(\min(\boldsymbol{\nu}, \vec{\mathbf{1}}), \vec{\mathbf{0}})$ to adjust $\boldsymbol{\nu}$ to the nearest vector in $[0,1]^n$, and $\overline{r}(x,\boldsymbol{\nu}) = \overline{r}(x, \phi(\boldsymbol{\nu})), \forall \boldsymbol{\nu} \in \mathbb{R}^n \setminus [0,1]^n$ to keep Lipschitz continuity throughout $\mathbb{R}^n$.

error of $\tilde{\mathbf{v}}(t)$ is bounded as

$$\|\tilde{\mathbf{v}}(t) - \mathbb{E}[\tilde{\mathbf{v}}(t)]\|_2 = \|M_\sigma^+ \vec{\mathbf{y}}(t) - M_\sigma^+ M_\sigma \boldsymbol{\nu}\|_2$$
$$= \left\| M_\sigma^+ \cdot \left[M_{x_1}\varepsilon(t_1); \cdots ; M_{x_{|\sigma|}}\varepsilon(t_{|\sigma|})\right]\right\|_2$$
$$= \left\| (M_\sigma^\intercal M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} M_{x_i}^\intercal M_{x_i}\varepsilon(t_i)\right\|_2$$
$$\le \max_{\boldsymbol{\nu}_0,\boldsymbol{\nu}_1,\cdots,\boldsymbol{\nu}_{|\sigma|}\in[0,1]^n} \left\| (M_\sigma^\intercal M_\sigma)^{-1}\sum_{i=1}^{|\sigma|} M_{x_i}^\intercal M_{x_i}(\boldsymbol{\nu}_i - \boldsymbol{\nu}_0)\right\|_2$$
$$\le \sqrt{n}\sum_{i=1}^{|\sigma|}\left\|(M_\sigma^\intercal M_\sigma)^{-1} M_{x_i}^\intercal M_{x_i}\right\|_2 .$$

Let $\beta_\sigma = \max \|(M_\sigma^\intercal M_\sigma)^{-1} \sum_{i=1}^{|\sigma|} M_{x_i}^\intercal M_{x_i}(\boldsymbol{\nu}_i - \boldsymbol{\nu}_0)\|_2$, where the max is taken from $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, \cdots, \boldsymbol{\nu}_{|\sigma|} \in [0,1]^n$. By the above derivation, we see that β_σ is a constant of the game instance independent of the outcome distribution p, and bounds the error of estimated outcome $\tilde{\mathbf{v}}(t)$. Henceforth, we use the simple error bound $\|\tilde{\mathbf{v}}(t) - \boldsymbol{\nu}\|_2 \le \beta_\sigma$ in the rest of this paper.

We present our Global Confidence Bound Algorithm (GCB) in Algorithm 1. Variable n_σ is a counter recording the number of rounds of exploration that have been played so far. GCB alternates between exploration and exploitation as follows. At the beginning it executes one round of exploration (in $|\sigma|$ time steps) to initialize the empirical mean of outcome vector $\hat{\boldsymbol{\nu}}$. Then at a time step $t \ge 1$, it computes action $\hat{x}$ that provides the best expected reward on $\hat{\boldsymbol{\nu}}$ (line 6) and $\hat{x}^-$ that provides the second best expected reward (line 7). We remark that these two lines rely on efficient problem-specific offline solutions that overcome the problem of exponentially many actions (See Section 5 for efficient solutions for the crowdsourcing application). Enumeration of all actions in $\mathcal{X}$ occurs nowhere else in the algorithm, and thus there is no dependency on the size of $\mathcal{X}$ in the running time of GCB other than lines 6 and 7.

Line 8 includes the key conditions deciding whether at time step t we should do exploration or exploitation. First, $\sqrt{\frac{\alpha f_{\mathcal{X}}(t)}{n_\sigma}}$ is a global confidence bound, where we set $\alpha = 24L^2\beta_\sigma^2$ and frequency function $f_{\mathcal{X}}(t) = \ln t + 2\ln|\mathcal{X}|$ for our analysis. This confidence bound is used to detect whether the estimated global optimal and second optimal solutions are significantly separated (first condition in line 8), and if so we are confident that we can exploit on the current optimal $\hat{x}$. Unlike UCB1 in (Auer et al., 2002), CUCB in (Chen et al., 2013) or CBP in (Bartók et al., 2012), our confidence bound is not on individual actions or base arms, but on the global optimal action. The second condition in line 8 is needed for our distribution-independent regret bound, and it guarantees that even if the difference of the estimated expected reward of the current best and second best actions remains very small, which could be true when there are multiple optimal actions, our algorithm will avoid endlessly repeating the exploration phase and will start exploitation if it knows that enough rounds of exploration have been done. The frequency function $f_{\mathcal{X}}(t)$ includes a term of $2\ln|\mathcal{X}|$, which typically does not appear in confidence bounds in online learning algorithms. This is another key aspect of our algorithm to make our regret dependent on $\ln|\mathcal{X}|$ instead of $|\mathcal{X}|$.

Algorithm 1 GCB

```
Require: $\sigma$, $\alpha$, $f_{\mathcal{X}}(t)$; $\forall x \in \mathcal{X}$, $\overline{r}(x,\cdot)$, $M_x$.
 1: Initialize $t \leftarrow 0$, $n_\sigma \leftarrow 0$
 2: loop
 3:   if $t = 0$ then
 4:     state ← begin_exploration                {initialize $\hat{\boldsymbol{\nu}}$}
 5:   else
 6:     $\hat{x} = \mathrm{argmax}_{x\in\mathcal{X}}\, \overline{r}(x, \hat{\boldsymbol{\nu}})$
 7:     $\hat{x}^- = \mathrm{argmax}_{x\in\mathcal{X}\setminus\{\hat{x}\}}\, \overline{r}(x, \hat{\boldsymbol{\nu}})$
 8:     if $\left(\overline{r}(\hat{x}, \hat{\boldsymbol{\nu}}) - \overline{r}(\hat{x}^-, \hat{\boldsymbol{\nu}}) > \sqrt{\frac{\alpha f_{\mathcal{X}}(t)}{n_\sigma}}\right)$
            or $\left(n_\sigma > t^{\frac{2}{3}} f_{\mathcal{X}}(t)\right)$ then
 9:       state ← exploitation
10:     else
11:       state ← begin_exploration
12:     end if
13:   end if
14:   if state = begin_exploration then
15:     {exploration phase:}
16:     for $s \leftarrow 1$; $s \le |\sigma|$; $s \leftarrow s + 1$ do
17:       play $x_s$ in observer set $\sigma$, and observe $\mathbf{y}_s$
18:       if $s = |\sigma|$ then
19:         $n_\sigma \leftarrow n_\sigma + 1$
20:         $\vec{\mathbf{y}}_{n_\sigma} = (\mathbf{y}_1; \mathbf{y}_2; \cdots ; \mathbf{y}_{|\sigma|})$
21:         $\tilde{\mathbf{v}}_{n_\sigma} = \mathbf{I}(M_\sigma, \vec{\mathbf{y}}_{n_\sigma})$      {estimate outcomes}
22:         $\hat{\boldsymbol{\nu}} = \frac{1}{n_\sigma}\sum_{j=1}^{n_\sigma} \tilde{\mathbf{v}}_j$              {take average}
23:       end if
24:       $t \leftarrow t + 1$
25:       state ← in_exploration
26:     end for
27:   else
28:     {exploitation phase:}
29:     play action $\hat{x}$
30:     $t \leftarrow t + 1$
31:   end if
32: end loop
```

In lines 15–26, GCB executes one round of exploration as a batch in $|\sigma|$ consecutive time steps, as we already explained. Lines 29–30 are for the simple exploitation phase, in which the current optimal action $\hat{x}$ is played. Note that GCB does not need feedback from playing $\hat{x}$. It only requires feedback for the actions in the global observer set, and the feedbacks for these actions are only required to guarantee that when stacking up their transformation matrices together the resulting matrix M_σ is of full column rank (Definition 3.1).

A few remarks about the algorithm are now in order. First, repeated exploration of a single global observer set σ is for the ease of algorithm presentation and regret analysis. In practice, one can find multiple global observer sets and explore them in order. The regret bound for such exploration is essentially the same.

Second, GCB requires efficient offline algorithms for both computing the best action and the second best action, given an expected outcome vector. It is reasonable to assume the existence of an efficient offline algorithm for computing the optimal solution, otherwise one cannot require sublinear regret that compares with the optimal action. For computing the second best action, one may be able to achieve it by disabling the best action in some way and then finding the best action in the remaining actions. (The example in Section 5 is done in this way.) Moreover, computing the second best action in line 7 can be replaced by a decision problem — deciding if there is an action besides $\hat{x}$ that has expected reward at least $\overline{r}(\hat{x}, \hat{\boldsymbol{\nu}}) - \sqrt{\frac{\alpha f_{\mathcal{X}}(t)}{n_\sigma}}$. In any case, these algorithms are problem-specific and require domain expertise on the application. GCB separates the concern of offline optimization from the online learning part. In contrast, other partial monitoring algorithms such as BALATON in (Bartók et al., 2011) and CBP in (Bartók et al., 2012) rely on explicit enumeration of all actions in their online learning part, and thus do not efficiently address the issue of exponentially large action space.

4.1. Results on Regret Bound

Recall the following problem-specific constants: the size of the global observer set $|\sigma|$; parameter L from the continuity assumption; error bound β_σ on estimated outcome; and the maximum difference in expected reward $R_{\max}$.

Theorem 4.1 and 4.2 provide the distribution-independent and distribution-dependent regret bounds for Algorithm 1.

Theorem 4.1 (Distribution-independent bound). *Let $f_{\mathcal{X}}(t) = \ln t + 2\ln|\mathcal{X}|$, and $\alpha = 24L^2\beta_\sigma^2$. The distribution-independent regret bound of Algorithm 1 is:*

$$\begin{aligned} R(T) \le & R_{\max}|\sigma| \cdot T^{\frac{2}{3}}(\ln T + 2\ln|\mathcal{X}|) + \frac{8}{3}L\beta_\sigma T^{\frac{2}{3}} \\ & + R_{\max}\left(|\sigma| + \frac{4e^2}{|\mathcal{X}|^4}\right). \end{aligned} \tag{3}$$

For distribution-dependent bound for an outcome distribution p with mean outcome vector $\boldsymbol{\nu}$, we define:

$$\Delta_x = \overline{r}(x^*, \boldsymbol{\nu}) - \overline{r}(x, \boldsymbol{\nu}), \tag{4}$$

$$\Delta_{\max} = \max\{\Delta_x : x \in \mathcal{X}\}, \tag{5}$$

$$\Delta_{\min} = \min\{\Delta_x : x \in \mathcal{X}, \Delta_x > 0\}. \tag{6}$$

Theorem 4.2 (Distribution-dependent bound). *Let $f_{\mathcal{X}}(t) = \ln t + 2\ln|\mathcal{X}|$, and $\alpha = 24L^2\beta_\sigma^2$. If the instance has a unique optimal action under outcome distribution p and mean outcome vector $\boldsymbol{\nu}$, the distribution-dependent regret bound of Algorithm 1 is:*

$$\begin{aligned} R(T) \le \sum_{x\in\sigma}\Delta_x \cdot \Bigg[& \frac{96L^2\beta_\sigma^2}{\Delta_{\min}^2}(\ln T + 2\ln|\mathcal{X}|) + \frac{4e^2}{|\mathcal{X}|^4}\ln T \\ & +1\Bigg] + \Delta_{\max}\cdot\left(\frac{3e^2}{|\mathcal{X}|^4} + \frac{941L^3\beta_\sigma^3}{\Delta_{\min}^3}\right). \end{aligned} \tag{7}$$

Theorem 4.1 shows that our algorithm achieves $O(T^{\frac{2}{3}}\ln T)$ distribution-independent regret, which is close to the theoretical bound of $\Theta(T^{\frac{2}{3}})$ for partial monitoring games with the global observability property (Antos et al., 2012). Theorem 4.2 shows that our algorithm achieves $O(\log T)$ distribution-dependent regret (assuming unique optimal action), which matches the theoretical lower bound for the classical MAB problem (Lai & Robbins, 1985).

From both the distribution-independent and distribution-dependent regrets, we see that the regret bounds depend linearly on $\ln|\mathcal{X}|$. When comparing with regret bounds of existing work on partial monitoring, such as Theorem 1 in (Bartók et al., 2012) for the CBP algorithm, we see that they have terms explicitly summing over all actions $x \in \mathcal{X}$, leading to regret bounds linearly dependent on $|\mathcal{X}|$. We are able to achieve this because we use a small global observer set σ (recall that $|\sigma| \le n$), and we include a term $2\ln|\mathcal{X}|$ in our frequency function $f_{\mathcal{X}}(t)$ so that we use more explorations to reduce the error probability in finding the optimal action $\hat{x}$, thus avoiding paying a regret linear in $|\mathcal{X}|$ for potentially exploiting a wrong $\hat{x}$.

In the distribution-dependent bound, we require that there is a unique optimal action in the problem instance. This is because of our weak assumption of the global observer set: we only assume feedbacks for the global observer set. To distinguish two actions we have to play the global observer set repeatedly and pay high regrets, but if the two are indeed both optimal we can never tell them apart and have to keep paying high regrets for exploration. In contrast, in MAB, CMAB and the more general partial monitoring games with the local observability property (Antos et al., 2012), the player is able to play multiple close-to-optimal actions for both exploration and exploitation purposes at the same time: if they are indeed all optimal, playing any of them is a good exploitation, and if they are different playing them will eventually tell them apart with a low regret.

To summarize, GCB algorithm employs the following features to achieve our goal: (a) GCB employs a global confidence bound condition, avoiding comparing the reward differences between all pairs of actions; (b) GCB separates the concern of offline optimization from online learning, and eliminates steps that require enumerating all actions from the online learning part; (c) GCB trades off more explorations for a lower error probability in exploitation to avoid the potential for such errors to incur a high regret cost linear to the action space size.

5. Crowdsourcing Application

In this section, we demonstrate our combinatorial partial monitoring model and the GCB algorithm using the following crowdsourcing application. Consider a crowdsourcing platform providing matchings between N workers and M tasks each day. For simplicity of description, we assume that $N \leq M$ and that these M tasks come from the same employer. Let random variable v_{ij} be the performance of worker i on task j. The performance could be measured as time to complete the task, proportion of the task completed, wage paid to the worker, or some performance evaluation value given by the employer. The actual form is unimportant and we assume it to be a continuous random variable normalized to the range $[0, 1]$. Therefore, the outcome vector can be represented by the random vector $\mathbf{v} = \left(v_{11}, v_{12}, \cdots, v_{1M}, v_{21}, \cdots, v_{NM}\right)^{\intercal} \in [0,1]^{NM}$. The action space is the set of matchings between workers and tasks in the complete bipartite graph connecting workers and tasks. In vector form, an action is a vector $\mathbf{x} = \left(z_{11}, z_{12}, \cdots, z_{1M}, z_{21} \cdots, z_{NM}\right)^{\intercal} \in \{0,1\}^{NM}$ with $\sum_{i=1}^{N} z_{ij} \leq 1$ and $\sum_{j=1}^{M} z_{ij} \leq 1$, where z_{ij} represents whether worker i and task j are matched. Thus, the size of the action space $\mathcal{X}$ is exponential in N and M.

The system's reward is defined as $r(\mathbf{x}, \mathbf{v}) = \mathbf{x}^{\intercal}\mathbf{v}$, i.e. the sum of performance of all matched worker-task pairs, which is often referred to as the social welfare. Because of linearity, we have $\overline{r}(\mathbf{x}, \boldsymbol{\nu}) = \mathbb{E}\left[r(\mathbf{x}, \mathbf{v})\right] = r(\mathbf{x}, \boldsymbol{\nu})$, where $\boldsymbol{\nu} = \mathbb{E}\left[\mathbf{v}\right]$, and the continuity assumption thus holds.

If we apply the CMAB framework to this problem, we would require feedback of daily performance for every worker-task pair. However, reporting daily performance for every worker-task pair is costly for the employer, and sometimes also raises privacy concerns, and thus the platform cannot expect to collect v_{ij} for all matched worker-task pairs every day. On the other hand, providing some feedback to the platform could help it to improve the matching in the future, which is beneficial to both employers and workers. Therefore, it is reasonable to assume that employers and workers would agree to provide partial feedback for at least a small set of matchings. For each matching that the platform receives feedback, we assume that the requested feedback from the employer consists of a single value, which is the aggregate reward of s matched worker-task pairs. With this setting, the transformation matrix $M_{\mathbf{x}}$ contains a single row with exactly s 1's and all other entries are 0, and $M_{\mathbf{x}}\mathbf{x} = s$.

It is easy to find a small global observer set such that their stacked matrix M_σ is of full column rank. For example, if $s = 1$, M_σ could simply be the identity matrix, which means that each action in the observer set is used for sampling the outcome of one action. If the application requires $1 < s < N$ to avoid revealing both individual workers' performance and the total performance received by the employer in a day, we can also construct the global observer set with a full-column-rank matrix M_σ. A simple construction is given in the supplementary material.

Therefore, we have modeled the crowdsourcing application as a problem instance in the framework of combinatorial partial monitoring with linear feedback, and we can apply our GCB algorithm to the application. For this application, the offline problem of finding the optimal action is equivalent to finding a matching that maximizes the total expected reward given the expected reward on each edge, which is exactly the maximum weighted matching problem, and thus can be efficiently solved. (The algorithm appears in many textbooks, e.g. (Kozen, 1992).) The offline problem of finding the second best matching can also be solved. We remove one edge in the optimal matching, and find the maximum weighted matching in the remaining graph. We then repeat this for all edges in the optimal matching and among the matchings computed select the one with the maximum weight. Therefore, both computing the best and the second best actions are efficient.

Hence, Theorems 4.1 and 4.2 apply to our application. The problem-specific constants are: $|\sigma| = NM$, $R_{\max} = N$, $L = \sqrt{N}$, and $\beta_\sigma = \sqrt{NM}$ for the case of $s = 1$. In summary, GCB can be applied to our crowdsourcing application setting, leading to low regret sublinear in T and polynomial in the offline problem instance size.

6. Future Work

Our work can be extended in several directions. One direction is to incorporate problems with only approximate algorithms for the offline optimization task, which is similar to the treatment in CMAB (Chen et al., 2013). Another direction is to extend the model to include more flexible observer sets, such that we could more tightly integrate exploration and exploitation. Moreover, how to efficiently compute the global observer set σ with both small $|\sigma|$ and β_σ in general needs to be addressed in the future.

References

Antos, András, Bartók, Gábor, Pál, Dávid, and Szepesvári, Csaba. Toward a classification of finite partial-monitoring games. *Theoretical Computer Science*, 2012.

Audibert, Jean-Yves and Bubeck, Sébastien. Minimax policies for adversarial and stochastic bandits. In *COLT*, 2009.

Auer, Peter, Cesa-Bianchi, Nicolò, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Bartók, G., Zolghadr, N., and Szepesvári, Cs. An adaptive algorithm for finite stochastic partial monitoring (extended version). In *ICML*, pp. 1–20, June 2012.

Bartók, Gábor, Pál, Dávid, and Szepesvári, Csaba. Minimax regret of finite partial-monitoring games in stochastic environments. *Journal of Machine Learning Research-Proceedings Track*, 19:133–154, 2011.

Bubeck, Sébastien and Cesa-Bianchi, Nicolò. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Cesa-Bianchi, Nicolo and Lugosi, Gábor. *Prediction, learning, and games*. Cambridge University Press, 2006.

Cesa-Bianchi, Nicolo and Lugosi, Gábor. Combinatorial bandits. *Journal of Computer and System Sciences*, 78 (5):1404–1422, 2012.

Cesa-Bianchi, Nicolo, Lugosi, Gábor, and Stoltz, Gilles. Regret minimization under partial monitoring. *Mathematics of Operations Research*, 31(3):562–580, 2006.

Chen, Wei, Wang, Yajun, and Yuan, Yang. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 151–159, 2013.

Gai, Yi, Krishnamachari, Bhaskar, and Jain, Rahul. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Trans. Netw.*, 20(5):1466–1478, October 2012. ISSN 1063-6692.

Kozen, Dexter. *The design and analysis of algorithms*. Springer, 1992.

Lai, Tze Leung and Robbins, Herbert. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Littlestone, Nick and Warmuth, Manfred K. The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pp. 256–261. IEEE, 1989.

Piccolboni, Antonio and Schindelhauer, Christian. Discrete prediction games with arbitrary feedback and loss. In *Computational Learning Theory*, pp. 208–223. Springer, 2001.

Robbins, Herbert. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pp. 169–177. Springer, 1985.

Vovk, Volodimir G. Aggregating strategies. In *Proc. Third Workshop on Computational Learning Theory*, pp. 371–383. Morgan Kaufmann, 1990.