Distributed Multi-Agent Online Learning Based on Global Feedback

Jie Xu, Cem Tekin, Simpson Zhang, and Mihaela van der Schaar

Abstract—**In many types of multi-agent systems, distributed agents cooperate with each other to take actions with the goal of maximizing an overall system reward. However, in many of these systems, agents only receive a (perhaps noisy) global feedback about the realized overall reward rather than individualized feedback about the relative merit of their own actions with respect to the overall reward. If the contribution of an agent's actions to the overall reward is unknown a priori, it is crucial for the agents to utilize a distributed algorithm that still allows them to learn their best actions. In this paper, we rigorously formalize this problem and develop online learning algorithms which enable the agents to cooperatively learn how to maximize the overall reward in these global feedback scenarios without exchanging any information among themselves. We prove that, if the agents observe the global feedback without errors, the distributed nature of the considered multi-agent system results in no performance loss compared with the case where agents can exchange information. When the agents' individual observations are erroneous, existing centralized algorithms, including popular ones like UCB1, break down. To address this challenge, we propose a novel class of distributed algorithms that are robust to individual observation errors and whose performance can be analytically bounded. We prove that our algorithms' learning regrets - the losses incurred by the algorithms due to uncertainty - are logarithmically increasing in time and thus the time average reward converges to the optimal average reward. Moreover, we also illustrate how the regret depends on the size of the action space, and we show that this relationship is influenced by the informativeness of the reward structure with regard to each agent's individual action. We prove that when the overall reward is fully informative, regret is linear in the total number of actions of all the agents. When the reward function is not informative, regret is linear in the number of joint actions. Our analytic and numerical results show that the proposed learning algorithms significantly outperform existing online learning solutions in terms of regret and learning speed. We illustrate how our theoretical framework can be used in practice by applying it to online Big Data mining using distributed classifiers. However, our framework can be applied to many other applications including online distributed decision making in cooperative multi-agent systems (e.g. packet routing or network coding in multi-hop networks), cross-layer optimization (e.g. parameter selection in different layers), multi-core processors etc.**

Index Terms—**Multi-agent learning, online learning, multi-armed bandits, Big Data mining, distributed cooperative learning, reward informativeness.**

I. Introduction

In this paper, we consider a multi-agent decision making and learning problem, in which a set of distributed agents select actions from their own action sets in order to maximize the overall system reward which depends on the joint action of all agents. In the considered scenario, agents do not know *a priori* how their actions influence the overall system reward, or how their influence may change dynamically over time. Therefore, in order to maximize the overall system reward, agents must dynamically learn how to select their best actions over time. But agents can only observe/measure the *overall* system performance and hence, they only obtain global feedback that depends on the joint actions of all agents. Since individualized feedback about individual actions is absent, it is impossible for the agents to learn how their actions alone affect the overall performance *without* cooperating with each other. However, because agents are distributed they are unable to communicate and coordinate their action choices. Moreover, agents' observations of the global feedback may be subject to individual errors, and thus it may be extremely difficult for an agent to conjecture other agents' actions based solely on its own observed reward history. The fact that individualized feedback is missing, communication is not possible, and the global feedback is noisy makes the development of efficient learning algorithms which maximize the joint reward very challenging. Importantly, the considered multi-agent learning scenario differs significantly from the existing solutions [12][20][21], in which agents receive individualized rewards. To help illustrate the differences, Figures 1(a) and (b) portray conventional learning in multi-agent systems based on individualized feedback and the considered learning in multi-agent systems based on global feedback with individual noise, respectively. The considered problem has many application scenarios. For instance, in a stream mining system that uses multiple classifiers for event detection in video streams, individual classifiers select operating points to classify specific objects or actions of interest, the results of which are synthesized to derive an event classification result. If the global feedback is only about whether the event classification is correct or not, individualized feedback about individual contribution is not available. For another instance, in a cooperative communication system, a set of wireless nodes forward signals of the same message copy to a destination node through noisy wireless channels. Each forwarding node selects its transmission scheme (e.g. power level) and the destination combines the forwarded signals to decode the original message using, e.g., a maximal ratio combination scheme. Since the message is only decoded using the combined signal but not individual signals, only a global reward depending on the joint effort of the forwarding nodes is available but not the nodes' individual contributions.

J. Xu, T. Cem and M. van der Schaar are with the Dept. of Electrical Engineering, University of California Los Angeles (UCLA). S. Zhang is with the Dept. of Economics, University of California Los Angeles (UCLA). This research is supported by the US Air Force Office of Scientific Research under the DDDAS Program.

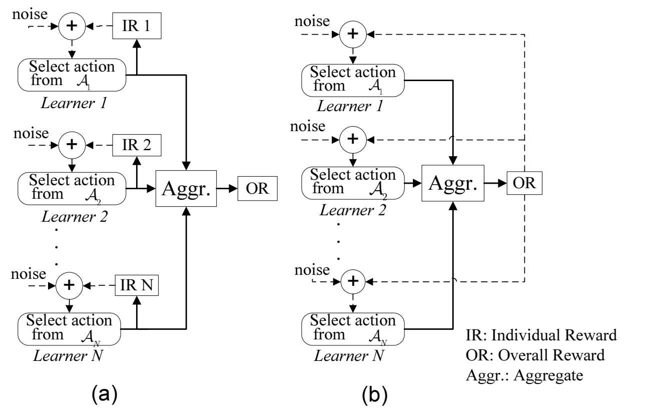


Fig. 1. Learning in multi-agent systems with (a) individualized feedback; (b) global feedback (this paper).

In this paper, we formalize for the first time the above multi-agent decision making framework and propose a systematic solution based on the theory of multi-armed bandits [9][10]. We propose multi-agent learning algorithms which enable the various agents to individually learn how to make decisions to maximize the overall system reward without exchanging information with other agents. In order to quantify the loss due to learning and operating in an unknown environment, we define the regret of an online learning algorithm for the set of agents as the difference between the expected reward of the best joint action of all agents and the expected reward of the algorithm used by the agents. We prove that, if the global feedback is received without errors by the agents, then all deterministic algorithms can be implemented in a distributed manner without message exchanges. This implies that the distributed nature of the system does not introduce any performance loss compared with a centralized system since there exist deterministic algorithms that are optimal. Subsequently, we show that if agents receive the global feedback *with* different (individual) errors, existing deterministic algorithms may break down and hence, there is a need for novel distributed algorithms that are robust to such errors. For this, we develop a class of algorithms which achieve a logarithmic upper bound on the regret, implying that the average reward converges to the optimal average reward[1]. The upper bound on regret also gives a lower bound on the convergence rate to the optimal average reward. For our first algorithm, DisCo, we start without any additional assumptions on the problem structure and show that the regret is still logarithmic in time. Although, the time order of the regret of the DisCo algorithm is logarithmic, due to its linear dependence on the cardinality of the joint action space, which increases exponentially with the number of agents, the regret is large and the convergence rate is very slow with many agents. Next, we define the informativeness of the overall reward function based on how effectively agents are able to distinguish the impact of their actions from the actions of others, and we exploit this informativeness in order to design improved learning algorithms. When the overall reward function is fully informative about the optimality of individual actions, the improved learning algorithm Disco-FI achieves a regret that is linear in the size of the action space of each agent, and logarithmic in time. The crucial idea behind this result is that, when the overall reward is fully informative, instead of using the *exact* reward estimates of every joint action, the agents can use the *relative* reward estimates of each individual action to learn their optimal actions at a much faster speed. Finally, we consider a more general setting where the global rewards are only partially informative. Our third algorithm (DisCo-PI) works when the overall reward function is informative for a group of agents instead of each agent individually, and it achieves a regret which is in between the first two algorithms. As an application of our theoretical framework, we then run simulations utilizing our algorithms for the problem of online Big Data mining using distributed classifiers [3][**?**][5][6]. We show that the proposed algorithms achieve a very high classification accuracy when compared with existing solutions. Our framework could also be similarly applied to many other applications including online distributed decision making in cooperative multi-agent systems such as multi-path or multi-hop networks, cross-layer design, multi-core processing systems, etc.

The remainder of this paper is organized as follows. In Section II, we describe the related work and highlight the differences from our work. In Section III, we build the system model and define the regret of a learning algorithm with respect to the optimal policy. In Section IV, we characterize the class of algorithms that can be implemented by distributed agents without errors, and we show that with errors centralized algorithms may break down. In Section V, the basic distributed cooperative learning algorithm is proposed and its regret performance is analyzed. Two improved learning algorithms, for fully informative and partially informative overall reward functions, are developed and analyzed in Sections VI and VII. Some discussions and extensions are provided in Section VIII. In Section IX, we evaluate the proposed algorithms through numerical results for the problem of Big Data stream mining. Finally, the concluding remarks are given in Section X.

II. Related Works

The literature on multi-armed bandit problems can be traced back to [7][8] which studies a Bayesian formulation and requires priors over the unknown distributions. In our paper, such information is not needed. A general policy based on upper confidence bounds is presented in [9] that achieves asymptotically logarithmic regret in time given that the rewards from each arm are drawn from an independent and identically distributed (i.i.d.) process. It also shows that no policy can do better than $\Omega(K \ln t)$ [2] (i.e. linear in the number of arms and logarithmic in time) and therefore, this policy is order optimal in terms of time. In [10], upper confidence bound (UCB) algorithms are presented which are proved to achieve logarithmic regret uniformly over time, rather than

[1]It is shown in [9] that logarithmic regret is the best possible even for simple single agent learning problems. However, convergence to the optimal reward is a much weaker result than logarithmic regret. Any algorithm with a sublinear bound on regret will converge to the optimal average reward asymptotically.

[2] We adopt the standard asymptotic notations $\Omega(\cdot)$ and $O(\cdot)$ as in [24].

only asymptotically. These policies are shown to be order optimal when the arm rewards are generated independently of each other. When the rewards are generated by a Markov process, algorithms with logarithmic regret with respect to the best static policy are proposed in [14] and [15]. However, all of these algorithms intrinsically assume that the reward process of each arm is independent, and hence they do not exploit any correlations that might be present between the rewards of different arms. In this paper the rewards may be highly correlated, and so it is important to design algorithms that take this into account.

Another interesting bandit problem, in which the goal is to exploit the correlations between the rewards, is the combinatorial bandit problem [23]. In this problem, the agent chooses an action vector and receives a reward which depends on some linear or non-linear combination of the individual rewards of the actions. In a combinatorial bandit problem the set of arms grows exponentially with the dimension of the action vector; thus standard bandit policies like the one in [10] will have a large regret. The idea in these problems is to exploit the correlations between the rewards of different arms to improve the learning rate and thereby reduce the regret [11][12]. Most of the works on combinatorial bandits assume that the expected reward of an arm is a linear function of the chosen actions for that arm. For example [13] assumes that after an action vector is selected, the individual rewards for each non-zero element of the action vector are revealed. Another work [22] considers combinatorial bandit problems with more general reward functions, defines the approximation regret and shows that it grows logarithmically in time. The approximation regret compares the performance of the learning algorithm with an oracle that acts approximately optimally, while we compare our algorithm with the optimal policy. This work also assumes that individual observations are available. However, in this paper we assume that only global feedback is available and individuals cannot observe each other's actions. Agents have to learn their optimal actions based only on the feedback about the overall reward. Other bandit problems which use linear reward models are studied in [14][15][16]. These consider the case where only the overall reward of the action profile is revealed but not the individual rewards of each action. However, our analysis is not restricted to linear reward models, but instead much more general. In addition, in most of the previous work on multi-armed bandits [9][10][11][12], the rewards of the actions (arms) are assumed to come from an unknown but fixed distribution. We also have this assumption in most of our analysis in this paper. However, in Section VII we propose learning algorithms which can be used when the distribution over rewards is changing over time (i.e. exhibits accuracy drift[3]).

Another line of work considers online optimization problems, where the goal is to minimize the loss due to learning the optimal vector of actions which maximizes the expected reward function. These works show sublinear (greater than logarithmic) regret bounds for linear or submodular expected reward functions, when the rewards are generated by an adversary to minimize the gain of the agent. The difference of our work is that we consider a more general reward function and prove logarithmic regret bounds. Recently, distributed bandit algorithms are developed in [28] in network settings. In that work, agents have the same set of arms with the same unknown distributions and are allowed to communicate with neighbors to share their observed rewards. In contrast, in the current paper, agents have distinct sets of arms, the reward depends on the joint action of agents and agents do not communicate at run-time.

Table I summarizes the comparison with existing works on bandits.

[3]Accuracy drift is more general than concept drift and is formally defined later in Section VIII.

	[9,10,11]	[12,17,18]	[13]	[22]	This work
Decentralized	No	Yes	No	No	Yes
Combinatorial with linear rewards	No	No	Yes	Yes	Yes
Combinatorial with nonlinear rewards	No	No	No	Yes	Yes
Only overall reward revealed	N/A	N/A	No	No	Yes
Exploit reward informativeness	No	No	No	No	Yes
Accuracy drift	No	No	No	No	Yes

TABLE I
COMPARISON WITH EXISTING WORKS ON BANDITS.

III. SYSTEM MODEL

There are N agents indexed by the set $\mathcal{N} = \{1, 2, ..., N\}$. Each agent has access to an action set $\mathcal{A}_n$, with the cardinality of the action set denoted by $K_n = |\mathcal{A}_n|$. Since we model the system using the multi-armed bandit framework, we will use "arm" and "action" interchangeably in this paper. In addition to the number of its own arms, each agent n knows the number of arms K_j of all the other agents $j \neq n$. The model is set in discrete time $t = 1, 2, ..., T$. In each time slot, each agent selects a single one of its own arms $a_n(t) \in \mathcal{A}_n$. Agents are distributed and thus cannot observe the arm selections of the other agents. We denote by $\boldsymbol{a}(t)$ the vector of arm selections by all the agents at time t, which we call the *joint arm* that is selected at time t.

Given any joint arm selection, a random reward $r_t(\boldsymbol{a}(t))$ will be generated according to an unknown distribution, with a dynamic range bounded by a value D'. For now, we will assume that this global reward is i.i.d. across time. We denote the expected reward given a selection $\boldsymbol{a}(t)$ by $\mu(\boldsymbol{a})=\mathbb{E}[r_t(\boldsymbol{a}(t))]$. The agents do not know the reward function $r_t(\boldsymbol{a}(t))$ initially and must learn it over time. Every period each agent n privately observes a signal $r_t^n = r_t + \epsilon_t^n$, equal to the global reward r_t plus a random noise term ϵ_t^n. We assume that ϵ_t^n has zero mean, is bounded in magnitude by D'' and i.i.d. across time, but it does not need to be i.i.d. across agents. Let $D = D' + D''$. Agents cannot communicate, so at any time t each agent has access to only its own history of noisy reward observations $\mathcal{H}_t^n = \{r_\tau^n\}_{\tau=1}^t$.

Agents operate according to an *algorithm* $\pi_n(\mathcal{H}_t^n)$, which tells it which arm to choose after every history of observations. This algorithm can be deterministic, meaning that given any history it will map to a unique arm, or probabilistic, meaning that for some histories it will map to a probability distribution over arms. Let $\pi(\mathcal{H}_t^1, ..., \mathcal{H}_t^N)=\{\pi_1(\mathcal{H}_t^1), ..., \pi_N(\mathcal{H}_t^N)\}$ denote the *joint algorithm* that is used by all agents after every possible history of observations. Since agents cannot communicate, the joint algorithm may only select actions for each agent based on that agent's private observation history. We denote the joint arm selected at time t given the joint algorithm as $\boldsymbol{a}^\pi(t)$. Fixing any joint algorithm, we can compute the expected reward at time 0 as $\mathbb{E}\sum_{t=1}^T r_t(\boldsymbol{a}^\pi(t))$.

This paper will propose a group of joint algorithms that can achieve sublinear regret in time given different restrictions on the expected reward function $\mu(\boldsymbol{a})$. Denote the optimal joint action by $\boldsymbol{a}^* := \arg\max_{\boldsymbol{a}} \mu(\boldsymbol{a})$. We will always assume that the optimal joint action is unique. The regret of a joint algorithm $\pi(\mathcal{H}_t^1, ..., \mathcal{H}_t^N)$ is given by

$$R(T;\pi) := T\mu(\boldsymbol{a}^*) - \mathbb{E}\sum_{t=1}^{T} r_t(\boldsymbol{a}^\pi(t)) \tag{1}$$

Regret gives the convergence rate of the total expected reward of the learning algorithm to the value of the optimal solution. Any algorithm whose regret is sublinear will converge to the optimal solution in terms of the average reward.

IV. Robustness of Algorithms with Distributed Implementation

In the considered setting, there is no individual reward observation associated with each individual arm but only an overall reward which depends on the arms selected by all agents. Therefore agents have to learn how their individual arm selections influence the overall reward, and choose the best joint set of arms in a cooperative but isolated manner. In general, agents may observe different noisy versions of the overall reward realization at each time, so we would like the algorithms to be robust to errors and perform efficiently in a noisy environment. But we will start by considering situations where there are no errors, and show that in this case agents are able to achieve the optimal expected reward even if they are distributed and unable to communicate.

A. Scenarios without individual observation errors

Let Π^c be the set of algorithms that can be implemented in a scenario where agents are allowed to exchange messages (reward observations, selected arms etc.) at run-time. Let Π^d be the set of algorithms that can be implemented in scenarios where agents cannot exchange messages at run-time. Obviously $\Pi^d \subseteq \Pi^c$. At the first sight, it seems that the restrictions on communication may result in efficiency loss compared to the scenario where agents can exchange messages. Next, we prove a perhaps surprising result – there is no efficiency loss even if agents cannot exchange messages at run-time as long as the agents observe the same overall reward realization in each time slot. Such a result is thus applicable if there are no errors, or even if the error terms, ϵ_t , are the same for every agent at every time t.

Theorem 1: If agents observe the same reward realization in each time slot, then $\min_{\pi\in\Pi^c} R(T;\pi) = \min_{\pi\in\Pi^d} R(T;\pi)$, $\forall T$.

Proof: See Appendix A in [31]. ■

Theorem 1 reveals that even if agents are distributed and not able to exchange messages at run-time, all existing deterministic algorithms proposed for centralized scenarios can still be used when agents observe the same reward realizations. The reason is that even though agents cannot directly communicate, as long as they know the algorithms of the other agents before runtime they can correctly predict which arms the other agents will choose based on the global reward history. In particular, the classic UCB1 algorithm can be implemented in distributed scenarios without loss of performance.

B. Scenarios with individual observation errors

When agents observe different noisy versions of the reward realizations, it is difficult for them to infer the correct actions of other agents based on their own private reward histories since their beliefs about others could be wrong and inconsistent. For instance, one agent may observe a high reward for a joint arm, while another agent observes a low reward. Then the first agent may decide to keep playing that joint arm, and believe that the other agent is also still playing it, while in actuality the other agent has already moved on to testing other joint arms. In such scenarios, even a single small observation error could cause inconsistent beliefs among agents and lead to error propagation that is never corrected in the future. In Proposition 1, we show this effect for the classic UCB1 algorithm and prove that UCB1 is not robust to errors when implemented in a distributed way.

Proposition 1: In distributed networks where agents do not exchange messages at run-time, if the observations of the overall reward realization are subject to individual errors, then the expected regret of the distributed version of UCB1 algorithm, in which each agent keeps an instance of UCB1 for its own actions and $N-1$ different instances of UCB1 for the actions of other agents, can be linear.

Proof: See Appendix B in [31]. ■

Proposition 1 implies that even if we implement existing deterministic algorithms such as UCB1 for distributed agents using the reward history, there is no guarantee on their performance when individual observation errors exist. Therefore, there is a need to develop new algorithms that are robust to errors in distributed scenarios. In the next few sections, we will propose such a class of algorithms that are robust to individual errors and can achieve logarithmic regret in time even in the noisy environment.

V. Distributed Cooperative Learning Algorithm

In this section, we propose the basic distributed cooperative learning (DisCo) algorithm which is suitable for any overall reward function. The proposed learning algorithm achieves logarithmic regret (i.e. $R(T) = O(\prod_{n=1}^N K_n \ln T)$). In Sections VI and VII, we will identify some useful reward structures and exploit them to design improved learning algorithms (DisCo-FI and DisCo-PI algorithms) which achieve even better regret results.

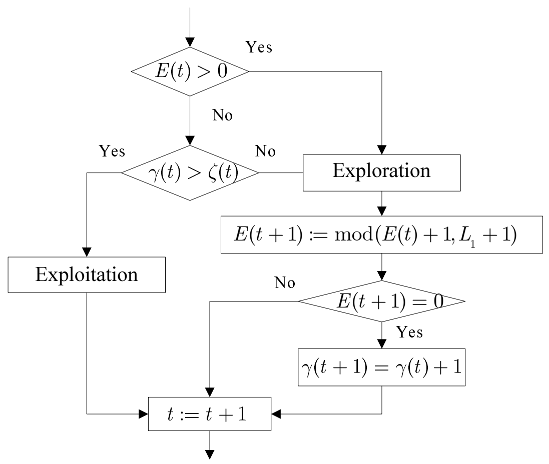


Fig. 2. Flowchart of the phase transition.

A. Description of the Algorithm

The DisCo algorithm is divided into phases: exploration and exploitation. Each agent using DisCo will alternate between these two phases, in a way that at any time t, either all agents are exploring or all are exploiting. In the exploration phase, each agent selects an arm only to learn about the effects on the expected reward, without considering reward maximization, and updates the reward estimates of the arm it selected. In the exploitation phase, each agent exploits the best (estimated) arm to maximize the overall reward.

Knowledge, Counters and Estimates: There is a deterministic control function $\zeta(t)$ of the form $\zeta(t) = A \ln t$ commonly known by all agents. This function will be designed and determined before run-time, and thus is an input of the algorithm. Each exploration phase has a fixed length of $L_1 = \prod_{n=1}^{N} K_n$ slots, equal to the total number of joint arms. Each agent maintains two counters [4]. The first counter $\gamma(t)$ records the number of exploration phases that they have experienced by time slot t. The second counter $E(t) \in \{0, 1, ..., L_1\}$ represents whether the current slot is an exploration slot and, if yes, which relative position it is at. Specifically, $E(t) = 0$ means that the current slot is an exploitation slot; $E(t) > 0$ means that the current slot is the $E(t)$-th slot in the current exploration phase. Both counters are initialized to zero: $\gamma(0) = 0, E(0) = 0$. Each agent n maintains L_1 sample mean reward estimates $\bar{r}^n(l)$ $\forall l \in \{1, ..., L_1\}$, one for each relative slot position in an exploration phase. Let b_l^n denote the arm selected by agent n in the l-th position in an exploration phase. These reward estimates are initialized to be $\bar{r}^n(l) = 0$ and will be updated over time using the realized rewards (the exact updating method will be explained shortly).

Phase Transition: Whether a new slot t is an exploration slot or an exploitation slot will be determined by the values of $\zeta(t), \gamma(t)$ and $E(t)$. At the beginning of each slot t, the agents first check the counter $E(t)$ to see whether they are still in the exploration phase: if $E(t) > 0$, then the slot is an exploration slot; if $E(t) = 0$, whether the slot is an exploration slot or an exploitation slot will then be determined by $\gamma(t)$ and $\zeta(t)$. If $\gamma(t) \leq \zeta(t)$, then the agents start a new exploration phase, and at this point $E(t)$ is set to be $E(t) = 1$. If $\gamma(t) > \zeta(t)$, then the slot is an exploitation slot. At the end of each exploration slot, counter $E(t+1)$ for the next slot is updated to be $E(t+1) \leftarrow mod(E(t)+1, L_1+1)$. When $E(t+1) = 0$, the current exploration phase ends, and hence the counter $\gamma(t+1)$ for the next slot is updated to be $\gamma(t+1) \leftarrow \gamma(t)+1$. Figure 2 provides the flowchart of the phase transition for the algorithm.

Prescribed Actions: The algorithm prescribes different actions for agents in different slots and in different phases.

(i) *Exploration phase*: As clear from the Phase Transition, an exploration phase consists of L_1 slots. In each phase, the agents select their own arms in such a way that every joint arm is selected exactly once. This is possible without communication if agents agree on a selection order for the joint arms before run-time. At the end of each exploration slot (the l^{th} slot), $\bar{r}^n(l)$ is updated to

$$\bar{r}^n(l) \leftarrow \frac{(\gamma(t) - 1)\bar{r}^n(l) + r_t^n}{\gamma(t)} \tag{2}$$

Note that the observed reward realization r_t^n at time t may be different for different agents due to errors.

(ii) *Exploitation phase*: Each exploitation phase has a variable length which depends on the control function $\zeta(t)$ and counter $\gamma(t)$. At each exploitation slot t, each agent n selects $a_n = \{b_{l^*}^n : l^* = \arg\max_l \bar{r}^n(l)\}$. That is, each agent n selects the arm with the best reward estimate among $\bar{r}^n(l)$, $\forall l \in \{1, ..., L_1\}$. Note that in the exploitation slots, an agent n does not need to know other agents' selected arms. Since agents have individual observation noises, it is also possible that l^* is different for different agents.

B. Analysis of the regret

At any exploitation slot, agents need sufficiently many reward observations from all sets of arms in order to estimate the best joint arm correctly with a probability high enough such that the expected number of mistakes is small. On the other hand, if the agents spend too much time in exploring, then the regret will be too large because they are not exploiting the best joint arm sufficiently often. The control function $\zeta(t)$ determines when the agents should explore and when they should exploit and hence balances exploration and exploitation. In Theorem 2, we will establish conditions on the control function $\zeta(t)$ such that the expected regret bound of the proposed DisCo algorithm is logarithmic in time. Let $\Delta^{max} = \max_{\boldsymbol{a} \neq \boldsymbol{a}^*} \{\mu(\boldsymbol{a}^*) - \mu(\boldsymbol{a})\}$ be the maximum reward loss by selecting any suboptimal joint arm, and let $\Delta^{min} = \min_{\boldsymbol{a} \neq \boldsymbol{a}^*}\{ \mu(\boldsymbol{a}^*) - \mu(\boldsymbol{a})\}$ be the reward difference between the best joint arm and the second-best joint arm.

Theorem 2: If $\zeta(t) = A \ln t$ with $A > 2\left(\frac{D}{\Delta^{min}}\right)^2$, then the expected regret of the DisCo algorithm after any number T periods is bounded by

$$R(T) \leq A L_1 \Delta^{max} \ln T + B_1 \tag{3}$$

where $B_1 = L_1 \Delta^{max} + \sum_t^{\infty} 2 N L_1 \Delta^{max} t^{-\frac{A}{2}\left(\frac{\Delta^{min}}{D}\right)^2}$ is a constant.

Proof: See Appendix C in [31]. ■

[4]Agents maintain these counters by themselves, $\gamma^n(t), E^n(t)$ $\forall n$. However, since agents update these counters in the same way, the superscript for the agent index is neglected in our analysis.

The regret bound proved in Theorem 2 is logarithmic in time which guarantees convergence in terms of the average reward, i.e. $\lim_{T\to\infty} \mathbb{E}[R(T)]/T = 0$. In fact, the order of the regret bound, i.e. $O(L_1 \ln T)$, is the lowest possible that can be achieved [9]. However, since the impact of individual arms on the overall (expected) reward is unknown and may be coupled in a complex way, it is necessary to explore *every* possible joint arm to learn its performance. This leads to a large constant that multiplies $\ln T$ which is on the order of $L_1 = \prod_{n=1}^{N} K_n$. If there are many agents, then $\prod_{n=1}^{N} K_n$ will be very large and hence, a large reward loss will be incurred in the exploration phases. This motivates us to design improved learning algorithms which do not require exploring all possible joint arms in order to improve the learning regret. In the next section, we will explore the informativeness (defined formally later) of the expected reward function to develop improved learning algorithms based on the basic DisCo algorithm. We first consider the best case (Full Informativeness) and then extend to the more general case (Partial Informativeness).

VI. A Learning Algorithm for Fully Informative Rewards

In many application scenarios, even if we do not know exactly how the actions of agents determine the expected overall rewards, some structural properties of the overall reward function may be known. For example, in the classification problem which uses multiple classifiers [**?**], the overall classification accuracy is increasing in each individual classifier's accuracy, even though each individual's optimal action is unknown a priori. Thus, some overall reward functions may provide higher levels of informativeness about the optimality of individual actions. In this section, we will develop learning algorithms that achieve improved regret results and faster learning speed by exploiting such information.

A. Reward Informativeness

We first define the informativeness of an expected overall reward function.

Definition 1: (*Informativeness*) An expected overall reward function $\mu(\boldsymbol{a})$ is said to be informative with respect to agent n if there exists a unique arm $a_n^* \in \mathcal{A}_n$ such that $\forall \boldsymbol{a}_{-n}$,, $a_n^* = \arg\max_{a_n} \mu(a_n, \boldsymbol{a}_{-n})$.
In words, if the reward is informative with respect to agent n, then for any choices of arms selected by other agents, agent n's best arms in terms of the expected overall reward is the same. Lemma 1 helps explain why such a reward function is "informative".

Lemma 1: Suppose that $\mu(a)$ is informative with respect to agent n and the unique optimal arm is a_n^*, then the following is true:

$$a_n^* = \arg\max_{a_n} \sum_{\boldsymbol{a}_{-n}} \theta_{\boldsymbol{a}_{-n}} \mu(a_n, \boldsymbol{a}_{-n}), \forall \theta_{\boldsymbol{a}_{-n}} \geq 0, \sum_{\boldsymbol{a}_{-n}} \theta_{\boldsymbol{a}_{-n}} = 1 \tag{4}$$

Proof: This is a direct result of the Definition 1. ■
Lemma 1 states that, for an agent n, the weighted average of the expected overall reward over all possible choices of arms by other agents is maximized at the optimal arm a_n^*. Moreover, the optimal arm is the same for all possible weights $\theta_{\boldsymbol{a}_{-n}}$, $\forall \boldsymbol{a}_{-n}$. It further implies that instead of using the *exact* expected overall reward estimate $\bar{r}^n(a_n, \boldsymbol{a}_{-n})$ to evaluate the optimality of an arm a_n, agent n can also use the *relative* overall reward estimate (i.e. the weighted average reward estimate). In this way, agent n needs to maintain only K_n relative overall reward estimates $\bar{r}^n(a_n)$ by selecting the arm a_n and can use these estimates to learn and select the optimal arm. In particular, let $w_{a_n,\boldsymbol{a}_{-n}}$ be the number of times that the joint arm $(a_n, \boldsymbol{a}_{-n})$ is selected in the exploration slots that are used to estimate $\bar{r}(a_n)$. Then,

$$\mathbb{E}\bar{r}^n(a_n) = \frac{\sum_{a_{-n}} w_{a_n,\boldsymbol{a}_{-n}} \mu(a_n, \boldsymbol{a}_{-n})}{\sum_{a_{-n}} w_{a_n,\boldsymbol{a}_{-n}}} \tag{5}$$

If $w_{a_n,\boldsymbol{a}_{-n}} = w_{\tilde{a}_n,\boldsymbol{a}_{-n}}, \forall a_n, \tilde{a}_n, \forall \boldsymbol{a}_{-n}$, then we have $\frac{w_{a_n,\boldsymbol{a}_{-n}}}{\sum_{a_{-n}} w_{a_n,\boldsymbol{a}_{-n}}} \triangleq \theta_{\boldsymbol{a}_{-n}}, \forall a_n$. Note that we don't need to know the exact value of $w_{a_n,\boldsymbol{a}_{-n}}$as long as $w_{a_n,\boldsymbol{a}_{-n}} = w_{\tilde{a}_n,\boldsymbol{a}_{-n}}, \forall a_n, \tilde{a}_n, \forall \boldsymbol{a}_{-n}$. Therefore, the relative reward estimates $\bar{r}^n(a_n)$ can be used to learn the optimal action a_n^* even if agents are not exactly sure what arms have been played by other agents.

Definition 2: (*Fully Informative*) An expected overall reward function $\mu(a)$ is said to be fully informative if it is informative with respect to all agents.

If the overall reward function is fully informative, then the agents only need to record the relative overall reward estimates instead of the exact overall reward estimates. Therefore, the key design problem of the learning algorithm is, for each agent n, to ensure that the weights in (4) are the same for the relative reward estimates of all its arms so that it is sufficient for agent n to learn the optimal arm using only these relative reward estimates.

We emphasize the importance of the weights $\theta_{\boldsymbol{a}_{-n}}, \forall \boldsymbol{a}_{-n}$ being the same for all $a_n \in \mathcal{A}_n$ of each agent n even though agent n does not need to know these weights exactly. If the weights are different for different a_n, then it is possible that $\bar{r}^n(a_n') > \bar{r}^n(a_n^*)$ merely because other agents are using their good arms when agent n is selecting a suboptimal arm a_n while other agents are using their bad arms when agent n is selecting the optimal arm a_n^*. Hence, simply relying on the relative reward estimates does not guarantee obtaining the correct information needed to find the optimal arm.

Reward functions that are fully informative exist for many applications. We identify a class of overall reward functions that are fully informative below.

Fully Informative Reward Functions: For each agent n, if there exists a function $f_n : \mathcal{A}_n \to \mathbb{R}$, such that for all joint arms $\boldsymbol{a}$, the expected reward can be expressed as a function $F : \mathbb{R}^N \to \mathbb{R}$ where $\mu(\boldsymbol{a}) = F(f_1(a_1), ..., f_N(a_N))$ and μ is monotone in $f_n, \forall f_{-n}, \forall n$, then $\mu(\boldsymbol{a})$ is fully informative.

We provide two concrete examples below.

(1) *Classification with multiple classifiers*. In the problem of event classification using multiple classifiers, each classifier is in charge of the classification problem of one specific feature of the target event [3][**?**][5][6]. The event is accurately classified if and only if all classifiers have their corresponding features classified correctly. Let $f_n(a_n)$ be the

unknown feature classification accuracy of classifier n by selecting the operating point a_n. Assuming that the features are independent, then the event classification accuracy can be expressed as $\mu(\boldsymbol{a}) = \prod_{n=1}^{N} f(a_n)$ given the selection of the joint operating points $\boldsymbol{a}$ of all classifiers. Hence the event classification accuracy is fully informative.

(2) *Network security monitoring using distributed learning*: A set of distributed learners (each in charge of monitoring a specific sub-network) make predictions about a potential security attack based on their own observed data (e.g. packets from different IP addresses to their corresponding sub-networks). Let the prediction of learner n be $\tilde{y}_n(t|a_n(t)) \in \{1, -1\}$ at time t by choosing a classification function a_n. Based on these predictions, an ensemble learner uses a weighted majority vote rule [27] to make the final prediction, i.e. $\hat{y}(t|\boldsymbol{a}(t)) = \text{sgn}(\boldsymbol{w} \cdot \tilde{\boldsymbol{y}}(t|a_n(t)))$, and takes the security measures accordingly. In the end, the distributed learners observe the outcome $r_n^t(\boldsymbol{a}(t))$ of the system which depends on the accuracy of the prediction, i.e. $|y(t) - \hat{y}(t|\boldsymbol{a}(t))|$ with $y(t)$ being the true security condition. Let $f_n(a_n) = \mathbb{E}\{|\tilde{y}_n(a_n) - \hat{y}|\}$ be the accuracy of learner n by choosing a classification function a_n. The reward function $\mu(\boldsymbol{a})$ is also monotone in $f_n(a_n)$ and hence is fully informative.

We note that in the first example different agents have orthogonal learning tasks (classification with respect to different features) while in the second example different agents have the same learning task (detecting the security attack). However, both examples exhibit the fully informative property and our proposed learning algorithms handle both cases effectively. The difference comes from the *speed of learning*. When agents have orthogonal learning tasks, they are more pivotal and so their actions have a greater influence on the rewards, which allows them to learn faster as well. This is highlighted in our simulation results in Section IX, where it is shown that when an agent becomes more pivotal it discovers its optimal action quicker.

B. Description of the Algorithm

In this subsection, we describe an improved learning algorithm. We call this new algorithm the DisCo-FI algorithm where "FI" stands for "Fully Informative" [5]. The key difference from the basic DisCo algorithm is that, in DisCo-FI, the agents will maintain relative reward estimates instead of the exact reward estimates.

Knowledge, Counters and Estimates: Agents know a common deterministic function $\zeta(t)$ and maintain two counters $\gamma(t)$ and $E(t)$. Now each exploration phase has a fixed length of $L_2 = \sum_{n=1}^{N} K_n$ slots and hence, $E(t) \in \{0, 1, ..., L_2\}$ with $E(t) = 0$ representing that the slot is an exploitation slot and $E(t) > 0$ representing that it is the $E(t)$-th relative slot in the current exploration phase. As before, both counters are initialized to be $\gamma(0) = 0, E(0) = 0$. Each agent n maintains K_n sample mean (relative) reward estimates $\bar{r}^n(a_n), \forall a_n \in \mathcal{A}_n$, one for each one of its own arms. These (relative) reward estimates are initialized to be $\bar{r}^n(a_n) = 0$ and will be updated over time using the realized rewards.

[5]The algorithm can run in the general case, but we bound its regret only when the overall reward function is fully informative.

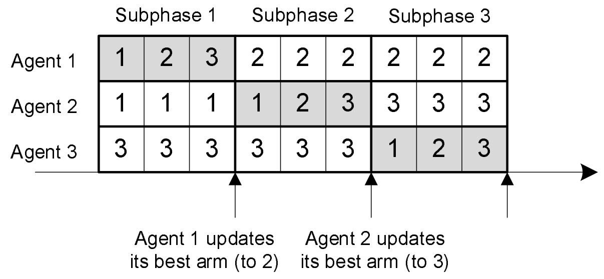


Fig. 3. Illustration of one exploration phase with 3 agents, each of which having 3 arms.

Phase Transition: The transition between exploration phases and exploitation phases are almost identical to that in the DisCo algorithm. The only difference is that at the end of each exploration slot, the counter $E(t+1)$ for the next slot is updated to be $E(t+1) \leftarrow mod(E(t)+1, L_2+1)$. Hence, we ensure that each exploration phase has only L_2 slots.

Prescribed Actions: The algorithm prescribes different actions for agents in different slots and in different phases.

(i) Exploration phase: As clear from the Phase Transition, an exploration phase consists of L_2 slots. These slots are further divided into N subphases and the length of the n^{th} subphase is K_n. In the n^{th} subphase, agents take actions as follows (Figure 3 provides an illustration):

1) Agent n selects each of its arms $a_n \in \mathcal{A}_n$ in turn, each arm for one slot. At the end of each slot in this subphase, it updates its reward estimate using the realized reward in this slot as follows,
$$\bar{r}^n(a_n) \leftarrow \frac{\gamma(t)\bar{r}^n(a_n) + r_t^n}{\gamma(t)+1} \tag{6}$$
2) Agent $i \neq n$ selects the arm with the highest reward estimate for every slot in this subphase, i.e. $a_i(t) = \arg\max_{a_i \in \mathcal{A}_i} \bar{r}^i(a_i)$.

(ii) Exploitation phase: Each exploitation phase has a variable length which depends on the control function $\zeta(t)$ and counter $\gamma(t)$. In each exploitation slot t, each agent n selects $a_n(t) = \arg\max_{a \in \mathcal{A}_n} \bar{r}^n(a)$.

C. Analysis of regret

We bound the regret of the DisCo-FI algorithm in Theorem 3. Let $\Delta_n^{min} = \min_{a_n \neq a_n^*, \boldsymbol{a}_{-n}} \{\mu(a_n^*, \boldsymbol{a}_{-n}) - \mu(a_n, \boldsymbol{a}_{-n})\}$ be the reward difference of agent n's best arm and its second-best arm, and let $\Delta_{FI}^{\min} = \min_n \Delta_n^{min}$.

Theorem 3: Suppose $\mu(\boldsymbol{a})$ is fully informative. If $\zeta(t) = A \ln T$ with $A \geq 2\left(\frac{D}{\Delta_{FI}^{\min}}\right)^2$, then the expected regret of the DisCo-FI algorithm after any number T slots is bounded by
$$R(T) < AL_2\Delta^{max} \ln T + B_2 \tag{7}$$
where $B_2 = L_2\Delta^{max} + 2L_2\Delta^{max} \sum_{t=1}^{\infty} t^{-\frac{A}{2}\left(\frac{\Delta_{FI}^{min}}{D}\right)^2}$ is a constant number.

Proof: See Appendix D in [31]. ■

The regret bound proved in Theorem 2 is also logarithmic in time for any finite time horizon T. Therefore, the average reward is guaranteed to converge to the optimal reward when the time horizon goes to infinity, i.e. $\lim_{T\to\infty} \mathbb{E}[R(T)]/T = 0$. Importantly, the proposed DisCo-FI algorithm exploits the informativeness of the expected overall reward function and achieves a much smaller constant that multiplies $\ln T$. Instead of learning every joint arm, agents can directly learn their own optimal arm through the relative reward estimates.

VII. A Learning Algorithm For Partially Informative Rewards

In the previous section, we developed the DisCo-FI algorithm for reward functions that are fully informative. However, in problems where the full informativeness property may not hold, the DisCo-FI algorithm cannot guarantee a logarithmic regret bound. In this section, we extend DisCo-FI to the more general case where the full informativeness constraint is relaxed. For example, in the classification problem which uses multiple classifiers, each classifier consists of multiple components each of which is considered as an independent agent. The accuracy of each individual classifier may depend on the configurations of these components in a complex way but the overall classification accuracy is still increasing in the accuracy of each individual classifier. Specifically, if the accuracy of one of these classifiers is increased, then the overall accuracy will increase independently of which configuration of the components of that classifier are chosen.

A. Partially Informativeness

We define an *agent group* and a *group partition* first.

Definition 3: (*Agent Group* and *Group partition*) An agent group g consists of a set of agents. A group partition $\mathcal{G} = \{g_1, ..., g_M\}$ of size M is a set of M agent groups such that each agent $n \in \mathcal{N}$ belongs to exactly one group.

We will call the set of arms selected by the agents in a group g_m a *group-joint arm* with respect to group g_m, denoted by $\boldsymbol{a}_m = \{a_n\}_{n\in g_m}$[6]. Denote the size of group g_m by N_m. It is clear that $\sum_{m=1}^{M} N_m = N$.

Definition 4: (*Group-Informativeness*) An expected overall reward function $\mu(\boldsymbol{a})$ is said to be informative with respect to a group g_m if there exists a unique group-joint arm $\boldsymbol{a}^*_m \in \times_{n\in g_m}\mathcal{A}_n$ such that $\forall \boldsymbol{a}_{-m}, \boldsymbol{a}^*_m = \arg\max_{\boldsymbol{a}_m\in\times_{i\in g_m}\mathcal{A}_i} \mu(\boldsymbol{a}_m,\ \mathrm{a}_{-m})$.

In words, for different choices of arms by other agents, group g_m's best group-joint arm is the same. Note that this is a generalization of Definition 1 because a group can also consist of only a single agent. Lemma 2 immediately follows.

Lemma 2: If $\mu(\boldsymbol{a})$ is informative with respect to an agent group g_m and the unique group-joint optimal arm is $\boldsymbol{a}^*_m$, then the following is true:

$$\begin{aligned} \boldsymbol{a}^*_m = \arg\max \textstyle\sum_{\boldsymbol{a}_{-m}} \theta_{\boldsymbol{a}_{-m}} \mu(\boldsymbol{a}_m, \boldsymbol{a}_{-m}), \\ \forall \theta_{\boldsymbol{a}_{-m}} \geq 0, \sum_{\boldsymbol{a}_{-m}} \theta_{\boldsymbol{a}_{-m}} = 1 \end{aligned} \tag{8}$$

Proof: This is a direct result of Definition 4. ■

[6]We abuse notation by using $\boldsymbol{a}_m$ $\boldsymbol{a}_{g_m}$. This should not introduce confusion given specific contexts.

Lemma 2 states that for an agent group g_m, the weighted average of the expected reward over all possible choices of arms by other agents is maximized at the optimal group-joint arm $\boldsymbol{a}_m$. Moreover, the optimal group-joint arm is the same for all possible weights. Therefore, to evaluate the optimality of a group-joint arm $\boldsymbol{a}_m$, the agents in group g_m ($\forall n \in g_m$) can use the relative reward estimate for that group-joint arm $\bar{r}^n(\boldsymbol{a}_m)$ instead of using the exact expected reward estimate $\bar{r}^n(\boldsymbol{a}_m, \boldsymbol{a}_{-m})$ as long as the weights $\theta_{\boldsymbol{a}_{-m}}, \forall \boldsymbol{a}_{-m}$, are the same for all $\boldsymbol{a}_m$.

Definition 5: (*Partially Informative*) An expected overall reward function $\mu(a)$ is said to be partially informative with respect to a group partition $\mathcal{G} = \{g_1, ..., g_M\}$ if it is informative with respect to all groups in $\mathcal{G}$.

Consider a surveillance problem in a wireless sensor network. Assume that there are multiple areas that are monitored by clusters of sensors. Let m be the m-th cluster of sensors. Each sensor selects a surveillance action. For instance, this action can be the position of the video camera, channel listened to by the sensor, etc. Let $\mu_m(\boldsymbol{a}_m)$ be the reward of the joint surveillance action taken by the sensors in cluster m. For example, this reward can be the probability of detecting an intruder that enters the area surveyed by the sensors in cluster m, Then depending on the strategic importance of these areas, the global reward is a linear combination of the rewards of the clusters, i.e., $\mu(\boldsymbol{a}) = \sum_m w_m \mu_w(\boldsymbol{a}_m)$. However, improving each individual sensor's action may not necessarily improve the accuracy of the cluster. In this case, the global reward is monotone in each cluster's reward but may not be monotone in each individual sensor's action. Thus, the reward function is partially informative.

If a reward function is fully informative, then it is also partially informative with respect to any group partition of the agents. On the other hand, if we take the entire agent set as one single group, then any reward function is partially informative with respect to this partition. Therefore, "Partially Informative" can apply to all possible reward functions through defining the group partition appropriately.

B. Description of the Algorithm

In this subsection, we propose the improved algorithm whose regret can be bounded for reward functions that are partially informative. We call this new algorithm the DisCo-PI algorithm where "PI" stands for "Partially Informative".

Knowledge, Counters and Estimates: Agents know a common deterministic function $\zeta(t)$ and maintain two counters $\gamma(t)$ and $E(t)$. In the DisCo-PI algorithm, each exploration phase has a fixed length of $L_3 = \sum_{m=1}^{M} \prod_{n\in g_m} K_n$ slots and hence, $E(t) \in \{0, 1, ..., L_3\}$ with $E(t) = 0$ representing that the slot is not an exploration slot and $E(t) > 0$ representing that it is the $E(t)$-th relative slot in the current exploration phase. Both counters are initialized to be $\gamma(0) = 0, E(0) = 0$. Each agent n in group g_m maintains $S_m = \prod_{i\in g_m} K_i$ reward estimates $\bar{r}^n(l), \forall l \in \{1, 2, ..., S_m\}$. Let $b^n_l \in \mathcal{A}_n$ denote the arm selected by agent n in the l^{th} slot in an exploration subphase. These (relative) reward estimates are initialized to be $\bar{r}^n(l) = 0$ and will be updated over time using the realized rewards.

Phase Transition: The algorithm works in a similar way as the first two algorithms in determining whether a slot is an exploration slot or an exploitation slot. The only difference is that the counter $E(t+1)$ is updated to be $E(t+1) \leftarrow mod(E(t)+1, L_3+1)$. This ensures that each exploration phase has L_3 slots.

Prescribed Actions: The algorithm prescribes different actions in different slots and in different phases.

(i) Exploration phase: An exploration phase consists of L_3 slots. These slots are further divided into M subphases and the length of the m^{th} subphase is $\prod_{n\in g_m} K_n$. In the m^{th} subphase, agents take actions as follows:

1) Agents in group g_m select the arms in such a way that every group-joint arm $\boldsymbol{a}_m$ with respect to group g_m is selected exactly once in this exploration subphase. At the end of the l^{th} slot in the exploration subphase, $\bar{r}^n(b_l^n)$ is updated to be

$$\bar{r}^n(l) \leftarrow \frac{\gamma(t)\bar{r}^n(l) + r_t^n}{\gamma(t)+1} \tag{9}$$

2) Agents i in group $g_j \neq g_m$ selects the component arm that forms the group-joint arm with the highest reward estimate, i.e. $a_i = \{b_{l^*}^i : l^* = \arg\max_l \bar{r}^i(l)\}$, for every slot in this subphase.

(ii) Exploitation phase: Each exploitation phase has a variable length which depends on the control function $\zeta(t)$ and counter $\gamma(t)$. In each exploitation slot t, each agent n of group g_m selects $a_n = \{b_{l^*}^n : l^* = \arg\max_l \bar{r}^n(l)\}$.

C. Analysis of regret

We bound the regret by running the DisCo-PI algorithm in Theorem 4. Let $\Delta_m^{min} = \min_{\boldsymbol{a}_m \neq \boldsymbol{a}_m, \boldsymbol{a}_{-m}} \{\mu(\boldsymbol{a}_m^*, \boldsymbol{a}_{-m}) - \mu(\boldsymbol{a}_m, \boldsymbol{a}_{-m})\}$ be the reward difference of the best group-joint arm of g_m and the second-best group-joint arm of g_m, and let $\Delta_{PI}^{min} = \min_m \Delta_m^{min}$.

Theorem 4: Suppose $\mu(\boldsymbol{a})$ is partially informative with respect to a group partition $\mathcal{G}$. If $\zeta(t) = A\ln T$ with $A \geq 2\left(\frac{D}{\Delta_{PI}^{min}}\right)^2$, then the expected regret of the DisCo-PI algorithm after any number T slots is bounded by

$$R(T) < AL_3\Delta^{max}\ln T + B_3 \tag{10}$$

where

$$B_3 = L_3\Delta^{max} + 2\sum_{m=1}^{M} N_m \prod_{n\in g_m} K_n \Delta^{max} \sum_{t=1}^{\infty} t^{-\frac{A}{2}\left(\frac{\Delta_{PI}^{min}}{D}\right)^2} \tag{11}$$

is a constant number.

Proof: See Appendix E in [31]. ∎

The regret bound proved in Theorem 4 is also logarithmic in time for any finite time horizon T. Therefore, the average reward is guaranteed to converge to the optimal reward when the time horizon goes to infinity, i.e. $\lim_{T\to\infty} \mathbb{E}[R(T)]/T = 0$. However, instead of learning every joint arm like in DisCo, agents in each group can learn just their own optimal group-joint arm using the relative reward estimates. Note that the constant that multiplies $\ln T$ is smaller than that of DisCo but larger than DisCo-FI. Table II summarizes the characteristics of the three proposed algorithms.

	DisCo	**DisCo-PI**	**DisCo-FI**
Reward Informativeness	Any	Partially Informative	Fully Informative
Learning Speed	Slow	Medium	Fast
Regret order	$O(\prod_{n=1}^{N} K_n \ln T)$	$O(\sum_{m=1}^{M}\prod_{n\in g_m} K_n \ln T)$	$O(\sum_{n=1}^{N} K_n \ln T)$

TABLE II
COMPARISON OF THE PROPOSED THREE ALGORITHMS.

VIII. EXTENSIONS

A. Missing and Delayed Feedback

In the previous analysis, we assumed that the global feedback is provided to the agents immediately at the end of each slot. In practice, this feedback may be missing or delayed and the delay may also be different for different agents since agents are distributed. We study the extension of our algorithm for these two scenarios in this subsection.

Consider the missing feedback case where the global feedback at time t may be missing with probability $1-p$. For instance, in the example of multiple classifiers, the ground truth label used to compute the global reward is not always available due to the high labeling cost and thus, the global reward may be sometimes unavailable. Our proposed algorithm can be easily extended as follows: whenever the feedback is missing in the exploration phase (which is observed by all agents), the agents repeat their current actions in the next slot until the feedback is received. Let $\bar{R}^{nm}(T)$ denote the regret bound of the proposed algorithms when there is no missing feedback and $\bar{R}^m(T)$ denote the regret bound of the modified algorithm with missing feedback by time T. Then we have the following proposition.

Proposition 2: $\bar{R}^m(T) = \bar{R}^{nm}(T) + \frac{p-1}{p}(A\ln T + 1)L\Delta^{max}$.

Proof: See Appendix F in [31]. ∎

Next, consider the delayed feedback case where the global feedback at time t arrives at time $t + L_n(t)$ for agent n, where $L_n(t)$is a random variable with support in $\{0, 1, ..., L_{max}\}$ and $L_{max} > 0$ is an integer. Our proposed algorithms can modified as follows: the exploration phase will be extended by L_{max} slots. The agents will update their reward estimates when the corresponding feedback is received. In each of the extended L_{max}slots, agents select the arms that maximize their estimated rewards in that slot. In this way, the algorithm ensures that sufficient labels have been received and hence, the reward estimates are sufficiently accurate in any exploitation slot. Let $\bar{R}^{nd}(T)$ denote the regret bound of the proposed algorithms with no delay and $\bar{R}^d(T)$ denote the regret bound of the modified algorithm with delays by time T. Then we have the following proposition.

Proposition 3: $\bar{R}^d(T) = \bar{R}^{nd}(T) + (A\ln T + 1)L_{max}\Delta^{max}$.

Proof: See Appendix G in [31]. ∎

In both cases with missing or delayed feedback, the modified algorithms achieve larger regrets than the original algorithms. However, the regret bounds are still logarithmic in time.

B. Accuracy drift

In the previous analysis, we assumed that even though the reward distributions and hence the expected rewards of the joint actions are unknown a priori, they are not changing over time. However, in some scenarios these rewards can be both unknown and time-varying due to changing system characteristics [7]. We refer to this as *accuracy drift*. In this case, the expected reward by selecting a joint arm a is also a time variable $\mu_t(a)$. Then the optimal joint arm is also a time variable, i.e. $\boldsymbol{a}^*(t) := \arg\max_{\boldsymbol{a}} \mu_t(\boldsymbol{a})$. The learning regret becomes $R(T) := \sum_{t=1}^{T} \mu_t(\boldsymbol{a}^*(t)) - \mathbb{E}\sum_{t=1}^{T} r_t(\boldsymbol{a}^{\pi}(t))$. Moreover, Δ^{max}, Δ^{min}, Δ^{min}_{FI}, Δ^{min}_{PI} will also be time variables $\Delta^{max}(t)$, $\Delta^{min}(t)$, $\Delta^{min}_{FI}(t)$, $\Delta^{min}_{PI}(t)$. We assume that Δ^{max} is upper bounded by $\bar{\Delta}^{max}(t)$ and $\Delta^{min}(t)$, $\Delta^{min}_{FI}(t)$, $\Delta^{min}_{PI}(t)$ are lower bounded by $\underline{\Delta}^{min}$.

Definition 6: (Accuracy Drift) The accuracy drift of the reward function for any two slots t, t' is defined to be $|\mu_t(\boldsymbol{a}) - \mu_{t'}(\boldsymbol{a})|$.

The proposed algorithms can be modified for use in deployment scenarios exhibiting accuracy drifts. Since the expected rewards are changing, using the realized rewards from the beginning of the system to estimate the expected rewards in the current slot will be very inaccurate. Therefore, agents should use only the most recent realized rewards to update the reward estimates. To do this, counter $\gamma(t)$ now maintains the number of exploration phases that have been experienced in the last Γ slots. The deterministic control function $\zeta(t)$ becomes a single control parameter which is independent of time, i.e. $\zeta(t) = A$. Whether a new slot is an exploration slot or an exploitation slot will still be determined by the values of $\zeta(t), \gamma(t)$ and $E(t)$ in a similar way to the previous algorithms. We bound the time-average regret in the following proposition.

Proposition 4: Suppose the accuracy drift satisfies $|\mu_t(\boldsymbol{a}) - \mu_{t'}(\boldsymbol{a})| \leq c_\Gamma, \forall t' : t \geq t' \geq t - \Gamma, \forall t$. If $\zeta(t) = A, (A < \Gamma/L)$, then the time-average expected regrets $\frac{R(T)}{T}$ of the modified algorithms after any number $T > \Gamma$ slots are bounded by

$$\begin{gathered}
\frac{AL_1 + 2(\Gamma - AL_1)NL_1 e^{-\frac{A}{2}\left(\frac{\underline{\Delta}^{min} - 2c_\Gamma}{D}\right)^2}}{\Gamma}\bar{\Delta}^{max} \\
\frac{AL_2 + 2(\Gamma - AL_2)L_2 e^{-\frac{A}{2}\left(\frac{\underline{\Delta}^{min}_{FI} - 2c_\Gamma}{D}\right)^2}}{\Gamma}\bar{\Delta}^{max} \\
\frac{AL_3 + 2(\Gamma - AL_3)\sum\limits_{m=1}^{M} N_m \prod\limits_{n=1}^{N} K_n e^{-\frac{A}{2}\left(\frac{\underline{\Delta}^{min}_{PI} - 2c_\Gamma}{D}\right)^2}}{\Gamma}\bar{\Delta}^{max}
\end{gathered} \tag{12}$$

for DisCo, DisCo-FI, DisCo-PI, respectively.

Proof: See Appendix H in [31]. ■

We note that the modified algorithms for accuracy drift do not achieve logarithmic regret in time since they have to track the changes in the expected reward by continuously exploring the arms at a constant rate. However, by exploiting the informativeness of the reward functions, better regret results can still be obtained by adopting DisCo-FI and DisCo-PI algorithms rather than the basic DisCo algorithm. Note that the bound given in Proposition 4 is not tight and the right hand side of the bound in (12) is time independent. When the right hand side of (12) is greater than D, this bound will not give us any information about the performance of the algorithm, since the maximum one-step loss of any algorithm is bounded by D. We see that the regret bound decays exponentially in the difference between $\underline{\Delta}^{min}$ and c_Γ, which is intuitive since a higher $\underline{\Delta}^{min} - 2c_\Gamma$ implies that it is easier to identify the best joint arm using only the observations in last Γ time steps.

[7]For example, in a Big Data system the distribution of the data stream may change which will result in a change in classification accuracies.

IX. Illustrative Results

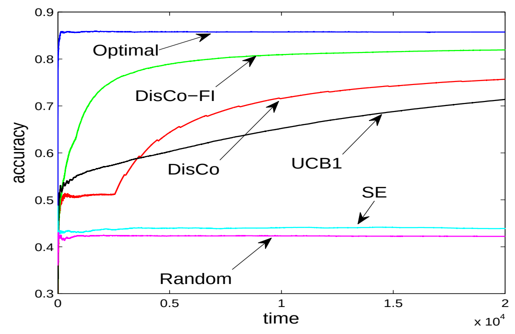
In this section, we illustrate the performance of the proposed learning algorithms via simulation results for the Big Data mining problem using multiple classifiers.

A. Big Data Mining using Multiple Classifiers

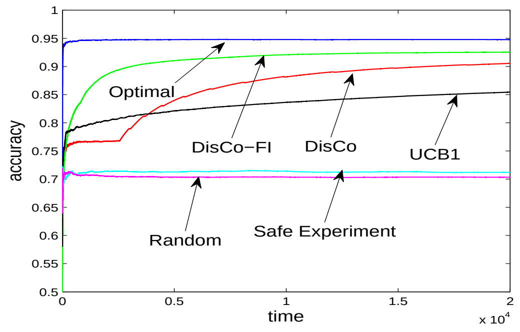
A plethora of online Big Data applications, such as video surveillance, traffic monitoring in a city, network security monitoring, social media analysis etc., require processing and analyzing streams of raw data to extract valuable information in real-time [1]. A key research challenge [26] in a real-time stream mining system is that the data may be gathered online by multiple distributed sources and subsequently it is locally processed and classified to extract knowledge and actionable intelligence, and then sent to a centralized entity which is in charge of making global decisions or predictions. The various local classifiers are not collocated and cannot communicate with each other due to the lack of a communication infrastructure (because of delays or other costs such as complexity [**?**][6]). Another stream mining problem may involve the processing of the same or multiple data stream, but require the use of classifier chains (rather than multiple single classifiers which are distributed as mentioned before) for its processing. For instance, video event detection [2][19] requires finding events of interest or abnormalities which could involve determining the concurrent occurrence (i.e. classification) of a set of basic objects and features (e.g. motion trajectories) by chaining together multiple classifiers which can jointly determine the presence of the event or phenomena of interest. The classifiers are often implemented at various locations to ensure scalability, reliability and low complexity [3][**?**]. For all incoming data, each classifier needs to select an operating point from its own set, whose accuracy and cost (e.g. delay) are unknown and may depend on the incoming data characteristics, in order to classify its corresponding feature and maximize the event classification accuracy (i.e. the overall system reward). Hence, classifiers need to learn from past data instances and the event classification performance to construct the optimal chain of classifiers. This classifier chain learning problem can be directly mapped into the considered multi-agent decision making and learning problem: agents are the component classifiers, actions are the operating points and the overall system reward is the event classification performance (i.e. accuracy minus cost).

B. Experiment Setup

Our proposed algorithm is tested using classifiers and videos provided by IBM's TRECVID 2007 project [25]. By extracting features such as color histogram, color correlogram, and co-occurrence texture, the classifiers are trained to detect high-level features, such as whether the video shot takes place



(a) Rule 1



(b) Rule 2

Fig. 4. Performance comparison for various algorithms.

outdoors or in an office building, or whether there is an animal or a car in the video. The classifiers are SVM-based and can therefore dynamically set detection thresholds for the output scores for each image without changing the underlying implementation. We chose this dataset due to the wide range of high-level features detected, which best models distributed classifiers trained across different sites. In the simulations, we use three classifiers (agents) to classify three features: (i) whether the image contained cars (CAR), (ii) whether the image contained mountains (MOU) (iii) whether the image is sports related (SPO). By synthesizing the feature classification results, the event detection result is obtained under two different rules.

1) **Rule 1**: The event is correctly classified if all three features are correctly classified.
2) **Rule 2**: The event is correctly classified if Feature 1 (CAR) is correctly classified and either Feature 2 (MOU) or Feature 3 (SPO) is correctly classified.

In the simulations, each classifier can choose from 4 operating points which will result in different accuracies. Let p_n denote the classification accuracy with respect to feature n. Assume that the classification of features is independent among classifiers, then the event classification accuracy p_{event} depends on the feature classification accuracy as follows:

$$\begin{aligned} p_{event} &= p_1 p_2 p_3 && \text{under Rule 1} \\ p_{event} &= p_1(1-(1-p_2)(1-p_3)) && \text{under Rule 2} \end{aligned} \tag{13}$$

Hence, the reward structure is fully informative for both event synthesis rules.

C. Performance Comparison

We implement the proposed algorithms and compare their performance against four benchmark schemes:

TABLE III
False Alarm and Miss Detection Rates.

	DisCo	DisCo-FI	UCB1	SE	Random
False Alarm	0.039	0.029	0.050	0.065	0.069
Miss Detection	0.249	0.194	0.356	0.469	0.496

(1) *Random*: In each period, each classifier randomly selects one operating point.

(2) *Safe Experimentation (SE)*: This is a method used in [5] when there is no uncertainty about the accuracy of the classifiers. In each period t, each classifier selects its baseline action with probability $1-\epsilon_t$ or selects a new random action with probability ϵ_t. When the realized reward is higher than the baseline reward, the classifiers update their baseline actions to the new action.

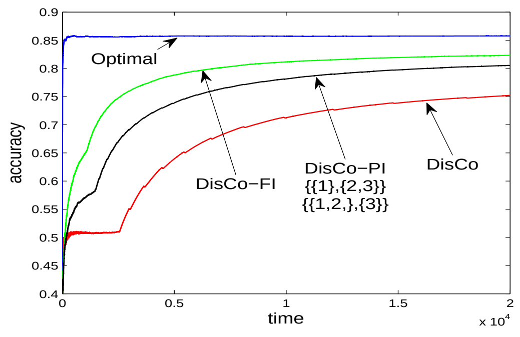
(3) *UCB1*: This is a classic multi-armed bandit algorithm proposed in [10]. As we showed in Proposition 1, there may be problems implementing this centralized algorithm in a distributed setting without message exchange. Nevertheless, for the sake of our simulations we will assume that there are no individual errors in the observation of the global feedback when we implement UCB1, and hence it can be perfectly implemented in our distributed environment.

(4) *Optimal*: In this benchmark, the classifiers choose the optimal joint operating points (trained offline) in all periods.

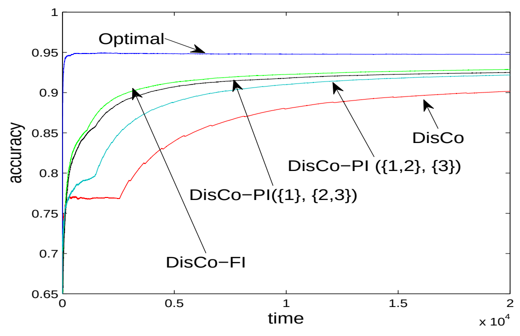
Figure 4 shows the achieved event classification accuracy over time under both rule 1 and rule 2. All curves are obtained by averaging 50 simulation runs. We also note that agents may receive noisy versions of the outcome (except for UCB1). Under both rules, SE works almost as poorly as the Random benchmark in terms of event detection accuracy. Due to the uncertainty in the detection results, updating the baseline action to a new action with a higher realized reward does not necessarily lead to selecting a better baseline action. Hence, SE is not able to learn the optimal operating points of the classifiers. UCB1 achieves a much higher accuracy than Random and SE algorithms and is able to learn the optimal joint operating points over time. However, the learning speed is slow because the joint arm space is large, i.e. $4^3 = 64$. The proposed DisCo algorithm can also learn the optimal joint action. However, since the joint arm space is large, the classifiers have to stay in the exploration phases for a relatively long time in the initial periods to gain sufficiently high confidence in reward estimates while the exploitation phases are rare and short. Thus, the classification accuracy is low initially. After the initial exploration phases, the classifiers begin to exploit and hence the average accuracy increases rapidly. Since the reward structure satisfies the Fully Informative condition, DisCo-FI rapidly learns the optimal joint action and performs the best among all schemes. Table III shows the false alarm and miss detection rates under rule 1 by treating one event as the null hypothesis and the remaining events as the alternative hypothesis.

D. Informativeness

Next, we compare the learning performance of the three proposed algorithms. For the DisCo-PI algorithm, we consider two group partitions $\{\{1\},\{2,3\}\}$ and $\{\{1,2\},\{3\}\}$. Figure 5



(a) Rule 1



(b) Rule 2

Fig. 5. Performance comparison for DisCo, DisCo-FI and DisCo-PI.

shows the learning performance over time for DisCo, DisCo-FI and DisCo-PI under Rule 1 and Rule 2. In both cases, DisCo-FI achieves the smallest learning regret and hence the fastest learning speed while the basic DisCo algorithm performs the worst. This is because DisCo-FI fully exploits the problem structure. The performance of the DisCo-PI algorithm is in between that of DisCo-FI and the basic DisCo algorithm. However, different group partitions have different impacts on the performance. Under Rule 1, the two group partitions perform similarly since the impacts of the three classifiers on the final classification result are symmetric. Under Rule 2, the impacts of classifier 2 and classifier 3 are coupled in a more complex way. Since the group partition $\{\{1\},\{2,3\}\}$ captures this coupling effect better, it performs better than the group partition $\{\{1,2\},\{3\}\}$. We note that even though that in this simulation DisCo-FI performs the best, in other scenarios where the reward function is only partially informative or even not informative, DisCo-PI and DisCo may perform better.

E. Impacts of reward function on learning speed

For both synthesis rules, the reward functions are fully informative, and so classifiers can learn their own optimal operating points using only the relative rewards. However, the same classifier will learn its optimal operating point at different speeds under different rules due to the differences in that classifier's impact on the global reward. Note that in the first rule, all the classifiers are processing different tasks of equal importance, whereas in the second rule classifiers 2 and 3 are less critical than classifier 1. Thus the learning speed for classifier 2 will be slower under the second rule because its impact is lower. This learning speed depends on the overall reward difference between the classifier's best operating point

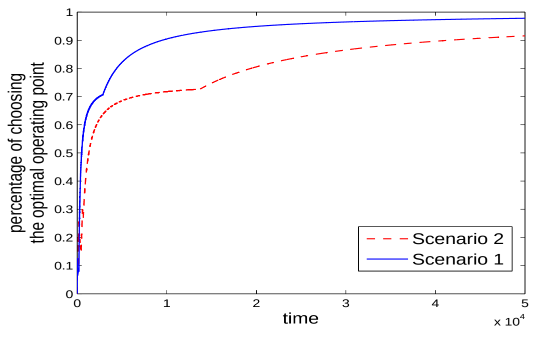
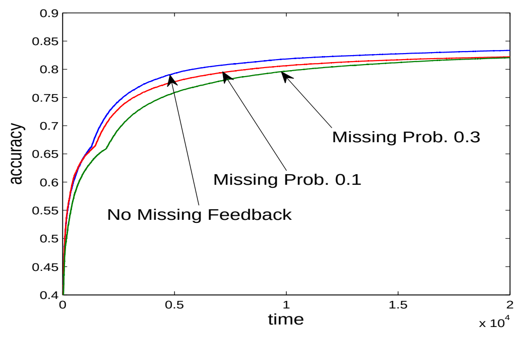
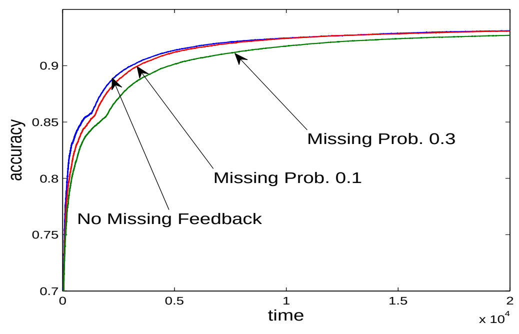


Fig. 6. Classifier 2 learns its optimal operating point at different speeds under different rules.



(a) Rule 1



(b) Rule 2

Fig. 7. Learning performance in scenarios with missing feedback.

and its second-best operating point, i.e. Δ_n^{min} . For classifier 2: under Rule 1, $\Delta_2^{min} = \min p_1 \min p_3 \Delta p_2$ and under Rule 2, $\Delta_2^{min} = \min p_1 \min(1-p_3)\Delta p_2$ with Δp_2 being the accuracy difference of classifier 2's best and second-best operating points. Since p_n is usually much larger than 0.5, Δ_2^{min} of Rule 2 is much smaller than that of Rule 1 and hence, classifier 2 learns its optimal operating point at a much slower speed under Rule 2 than Rule 1. Figure 6 illustrates the percentage of choosing the optimal operating point by classifier 2 under different rules.

F. Missing and Delayed Feedback

In this set of simulations, we study the impact of missing and delayed global feedback on the learning performance of the proposed algorithm. In Figure 7, we show the accumulating accuracy of the modified DisCo-FI algorithm for three missing feedback scenarios – there is no missing feedback, the missing probability is 0.1 and 0.3. A larger missing probability induces lower classification accuracy for a given time. Nevertheless, the proposed algorithm is not very sensitive to missing feedbacks. Even if the missing probability is relatively large, the degradation of the learning performance is small.

In Figure 8, we show the accumulating accuracy of the modified DisCo-FI algorithm for three delayed feedback scenarios – there is no delay, the maximal delay is 50 slots and 100 slots. Under both synthesis rules, learning is the fastest without feedback delays, and the larger the delay, the slower the learning speed. However, even with delays, the proposed DisCo-FI algorithm is still able to achieve logarithmic regret.

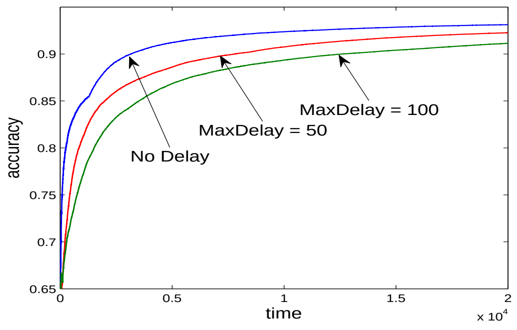
G. Accuracy Drift

Finally, we study the impact of accuracy drift on the learning performance of the proposed algorithm. In this set of simulations, the accuracies of the operating points of the classifiers change every fixed number of periods [8]. Figure 9 shows the accumulating average accuracy over time for DisCo-FI under Rule 1 and Rule 2, with the period length being 3000 slots, 1000 slots and 500 slots. The longer the period length, the smaller the accuracy drift. If the period length is infinity, then the expected accuracy is static and hence there is no accuracy drift. Several observations are worth noting. First, the learning performance is better if the accuracy drift is smaller. However, in all cases, the learning accuracy is almost constant and does not approach the optimal accuracy over time. This is because in order to track the changes in the accuracy, the algorithm uses only the recent reward feedbacks to estimate the accuracies of the operating points. Hence, the estimation error does not diminish as times increases. Second, the degree of impact of the accuracy drift depends on the reward functions. In this simulation, the reward function induced by Rule 2 is less vulnerable to accuracy drift. Third, we also note that the learning accuracy for the first period is relatively higher than that for the later periods. This is because the initial reward estimates are less corrupted by using wrong reward feedbacks and hence, they are more accurate.

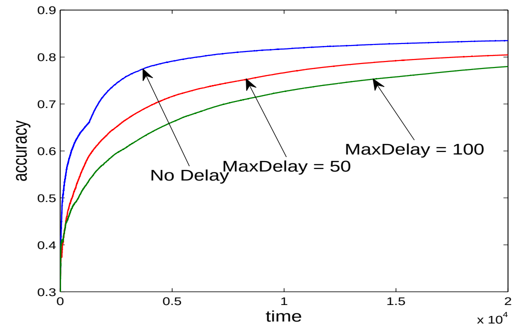
X. Conclusions

In this paper, we studied a general multi-agent decision making problem in which decentralized agents learn their best actions to maximize the system reward using only noisy observations of the overall reward. The challenging part is that individualized feedback is missing, communication among agents is impossible and the global feedback is subject to individual observation errors. We proposed a class of distributed cooperative learning algorithms that addresses all these problems. These algorithms were proved to be able to achieve logarithmic regret in time. We also proved that by exploiting the informativeness of the reward function, much better regret results can be achieved by our algorithms compared with existing solutions. Through simulations we applied the proposed learning algorithms to Big Data stream mining problems and showed significant performance improvements. Importantly, our theoretical framework can also be applied to learning in other types of multi-agent systems where communication between agents is not possible and agents observe only noisy global feedback.

[8]This is realized by swapping the indices of the operating points for each classifier.

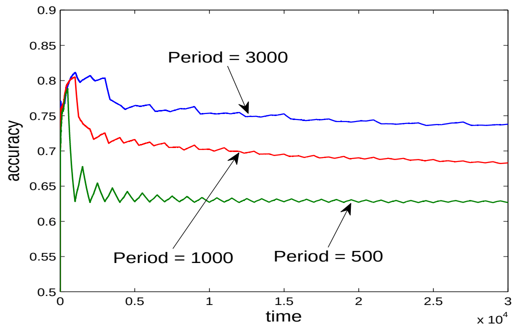


(a) Rule 1

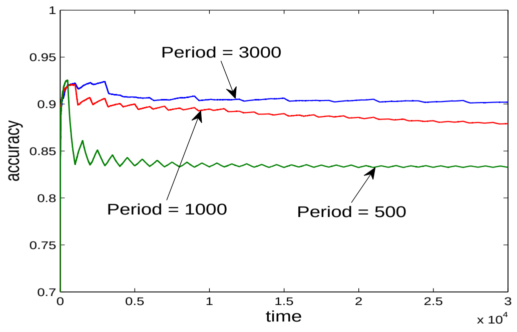


(b) Rule 2

Fig. 8. Learning performance in scenarios with delayed feedback.



(a) Rule 1



(b) Rule 2

Fig. 9. Learning performance in scenarios with accuracy drift.

References

[1] M. Shah, J. Hellerstein, M. Franklin, “Flux: An adaptive partitioning operator for continuous query systems,” *International Conference on Data Engineering (ICDE)*, 2003.

[2] Y. Jiang, S. Bhattacharya, S. Chang, and M. Shah, “High-level event recognition in unconstrained videos,” *Int. J. Multimed. Info. Retr.*, Nov. 2013

[3] F. Fu, D. Turaga, O. Verscheure, and M. van der Schaar, “Configuring competing classifier chains in distributed stream mining systems,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, Dec 2007.

[4] J. Xu, C. Tekin and M. van der Schaar, “Learning optimal classifier chains for real-time big data mining,” *51st Annual Allerton Conference on Communication, Control and Computing*, 2013.

[5] B. Foo and M. van der Schaar, “A rules-based approach for configuring chains of classifiers in real-time stream mining systems,” *EURASIP Journal on Advances in Signal Processing*, vol. 2009, 2009.

[6] R. Ducasse, D. Turaga, and M. van der Schaar, “Adaptive topologic optimization for large-scale stream mining,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 3, June 2010.

[7] Gittins, John C, “Bandit processes and dynamic allocation indices,” *Journal of the Royal Statistical Society. Series B (Methodological)* (1979): 148-177.

[8] P. Whittle, “Multi-armed bandits and the Gittins index,” *Journal of the Royal Statistical Society, Series B (Methodological)*, 1980.

[9] T. Lai and H. Robbins, “Asymptotically efficient adaptive allocation rules,” *Adv. Appl. Math.*, vol. 6, no. 1, 1985.

[10] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Mach. Learn.*, vol. 47, no. 2-3, 2002.

[11] V. Anantharam, P. Varaiya, and J. Walrand, “Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays C Part I: I. I. D. reward,” *IEEE Trans. Autom. Control*, vol. AC-32, no. 11, 1987.

[12] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami, “Distributed algorithms for learning and cognitive medium access with logarithmic regret,” *IEEE J. Sel. Areas Commun.*, vol. 29, no. 4, Apr. 2011.

[13] Y. Gai, B. Krishnamachari, and R. Jain, “Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations,” *IEEE/ACM Trans. Networking*, vol. 20, no. 5, 2012.

[14] P. Rusmevichientong and J. N. Tsitsiklis, “Linearly parameterized bandits,” *Math. Oper. Res.*, vol. 35, no. 2, 2010.

[15] P. Auer, “Using confidence bounds for exploitation-exploration tradeoffs,” *J. Mach. Learn. Res.*, vol. 3, pp. 397-422, 2002.

[16] V. Dani, T. P. Hayes, and S. Kakade, “Stochastic linear optimization under bandit feedback,” in *Proc. 21st Annu. COLT*, 2008.

[17] C. Tekin and M. Liu, “Online learning of rested and restless bandits,” *IEEE Trans. on Info. Theory*, vol. 58, no. 5, pp. 5588-5611, 2012.

[18] H. Liu, K. Liu and Q. Zhao, “Learning in a changing world: restless multi-armed bandit with unknown dynamics,” *IEEE Trans. on Info. Theory*, vol. 59, no. 3, pp. 1902-1916, 2012.

[19] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, “Enhancing video concept detection with the use of tomographs,” in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2013.

[20] C. Tekin and M. Liu, “Performance and convergence of multi-user online learning and its application in dynamic spectrum sharing,” in *Mechanisms and Games for Dynamic Spectrum Allocation*, Cambridge University Press, available from January 2014.

[21] K. Liu and Q. Zhao, “Distributed learning in multi-armed bandit with multiple players,” *IEEE Trans. Signal Process.*, vol. 58, no. 11, Nov. 2010.

[22] W. Chen, Y. Wang and Y. Yuan, “Combinatorial multi-armed bandit: general framework and applications,” in *Proc. 30th Intl. Conf. Machine Learning (ICML-13)*, pp. 151-159, 2013.

[23] N. Cesa-Bianchi, G. Lugosi, “Combinatorial bandits,” *J. Comp. Sys.* Sci. 78, no. 5, pp. 1404-1422, 2012.

[24] C.E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, Ed. T. H. Cormen, he MIT press, 2001.

[25] M. Campbell, A. Haubold, M. Liu, et al., “IBM Research TRECVID-2007 Video Retrieval System”.

[26] R. Ducasse and M. van der Schaar, “Finding It Now: Construction and Configuration of Networked Classifiers in Real-Time Stream Mining Systems,” *Handbook of Signal Processing Systems*, Springer New York, Ed. S. S. Bhattacharyya, F. Deprettere, R. Leupers and J. Takala, 2013.

[27] N. Littleston and M. K. Warmuth, “The weighted majority algorithm,” *Inf. Comput.*, vol. 108, no. 2, pp. 212-261, Feb. 1994.

[28] B. Szorenyi et al. “Gossip-based distributed stochastic bandit algorithms,” *Proc. 30th International Conference on Machine Learning*, 2013.

[29] J. Xu, M. van der Schaar, J. Liu and H. Li, “Forecasting popularity of videos using social media,”, *IEEE J. Sel. Topics Signal Process. (forthcoming)*, Nov., 2014.

[30] J. Xu, D. Deng, U. Demiryurek, C. Shahabi and M. van der Schaar, “Mining the situation: spatiotemporal traffic prediction with big data,” *IEEE J. Sel. Topics Signal Process. (forthcoming)*, 2015.

[31] Online appendix http://medianetlab.ee.ucla.edu/papers/XuTSP15App.pdf.