Online Team Formation Under Different Synergies

Matthew Eichhorn[1(✉)], Siddhartha Banerjee[1], and David Kempe[2]

[1] Cornell University, Ithaca, NY 14850, USA
{meichhorn,sbanerjee}@cornell.edu
[2] University of Southern California, Los Angeles, CA 90089, USA

Abstract. Team formation is ubiquitous in many sectors: education, labor markets, sports, etc. A team's success depends on its members' latent types, which are not directly observable but can be (partially) inferred from past performances. From the viewpoint of a principal trying to select teams, this leads to a natural exploration-exploitation trade-off: retain successful teams that are discovered early, or reassign agents to learn more about their types? We study a natural model for online team formation, where a principal repeatedly partitions a group of agents into teams. Agents have binary latent types, each team comprises two members, and a team's performance is a symmetric function of its members' types. Over multiple rounds, the principal selects matchings over agents and incurs regret equal to the deficit in the number of successful teams versus the optimal matching for the given function. Our work provides a complete characterization of the regret landscape for all symmetric functions of two binary inputs. In particular, we develop team-selection policies that, despite being agnostic of model parameters, achieve optimal or near-optimal regret against an adaptive adversary.

Keywords: Online team formation · Regret · Combinatorial bandits

1 Introduction

An instructor teaching a large online course wants to pair up students for assignments. The instructor knows that a team performs well as long as at least one of its members has some past experience with coding, but unfortunately, there is no available information on the students' prior experience. However, the course staff can observe the *performance* of each team on assignments, and so, over multiple assignments, would like to reshuffle teams to try and quickly maximize the overall number of successful teams. How well can one do in such a situation?

Team formation is ubiquitous across many domains: homework groups in large courses, workers assigned to projects on online labor platforms, police officers paired up for patrols, athletes assigned to teams, etc. Such teams must often be formed without prior information on each individual's latent skills or

Extended version with proofs can be found at https://arxiv.org/abs/2210.05795.

K. A. Hansen et al. (Eds.): WINE 2022, LNCS 13778, pp. 78–95, 2022.
https://doi.org/10.1007/978-3-031-22832-2_5

personality traits, albeit with knowledge of how these latent traits affect team performance. The lack of information necessitates a natural trade-off: a principal must decide whether to exploit successful teams located early or reassign teammates to gain insight into the abilities of other individuals. The latter choice may temporarily reduce the overall rate of success.

To study this problem, we consider a setting (described in detail in Sect. 2) where agents have binary latent types, each team comprises two members, and the performance of each team is given by the same *synergy function*, i.e., some given symmetric function of its members' types. Over multiple rounds, the principal selects matchings over agents, with the goal of minimizing the cumulative *regret*, i.e., the difference between the number of successful teams in a round versus the number of successful teams under an optimal matching. Our main results concern the special case of symmetric Boolean synergy functions—in particular, we study the functions EQ and XOR (in Sect. 3), OR (in Sect. 4) and AND (in Sect. 5). While this may at first appear to be a limited class of synergy functions, in Sect. 2.3, we argue that these four functions are in a sense the *atomic primitives* for this problem; our results for these four settings are sufficient to handle *arbitrary symmetric synergy functions*.

The above model was first introduced by Johari et al. [12], who considered the case where agent types are i.i.d. Bernoulli(p) (for known p) and provide asymptotically optimal regret guarantees under AND (and preliminary results for OR). As with any bandit setting, it is natural to ask whether one can go beyond a stochastic model to admit *adversarial* inputs. In particular, the strongest adversary one can consider here is an *adaptive adversary*, which observes the choice of teams in each round, and only then fixes the latent types of agents. In most bandit settings, such an adversary is too strong to get any meaningful guarantees; among other things, adaptivity precludes the use of randomization as an algorithmic tool, and typically results in every policy being as bad as any other. Nevertheless, in this work, we provide a *near-complete characterization of the regret landscape for team formation under an adaptive adversary*. In particular, in a setting with n agents of which k have type '1', we present algorithms that are agnostic of the parameter k, and yet when faced with an adaptive adversary, achieve optimal regret for EQ and XOR, and near-optimal regret bounds under OR and AND (and therefore, using our reduction in Sect. 2.3, achieve near-optimal regret for any symmetric function).

While our results are specific to particulars of the model, they exhibit several noteworthy features. First, despite the adversary being fully adaptive, our regret bounds differ only by a small constant factor from prior results for AND under i.i.d. Bernoulli types [12]; such a small gap between stochastic and adversarial bandit models is uncommon and surprising. Next, our bounds under different synergy functions highlight the critical role of these functions in determining the regret landscape. Additionally, our algorithms expose a sharp contrast between learning and regret minimization in our setting: while the rate of learning increases with more exploration, minimizing regret benefits from maximal exploitation. Finally, to deal with adaptive adversaries in our model, we use

techniques from extremal graph theory that are atypical in regret minimization; we hope that these ideas prove useful in other complex bandit settings.

1.1 Related Work

Regret minimization in team formation, although reminiscent of *combinatorial bandits/semi-bandits* [4–6,8,16], poses fundamentally new challenges arising from different synergy functions. In particular, a crucial aspect of bandit models is that rewards and/or feedback are linear functions of individual arms' latent types. Some models allow rewards/feedback to be given by a non-linear *link* function of the sum of arm rewards [7,10], but typically require the link function to be well-approximated by a linear function [17]. In contrast, our team synergy functions are *non-linear*, and moreover, are not well-approximated by any non-linear function of the sums of the agents' types.

One way to go beyond semi-bandit models and incorporate pairwise interactions is by assuming that the resulting reward matrix is low-rank [13,20,24]. The critical property here is that under perfect feedback, one can learn all agent types via a few 'orthogonal' explorations; this is true in our setting under the XOR function (Sect. 3), but not for other Boolean functions. Another approach for handling complex rewards/feedback is via a Bayesian heuristic such as Thompson sampling or information-directed sampling [9,14,19,23]. While such approaches achieve near-optimal regret in many settings, the challenge in our setting is in updating priors over agents' types given team scores. We hope that the new approaches we introduce could, in the future, be combined with low-rank decomposition and sampling approaches to handle more complex scenarios such as shifting types and corrupted feedback.

In addition to the bandit literature, there is a parallel stream on learning for team formation. Rajkumar et al. [18] consider the problem of learning to partition workers into teams, where team compatibility depends on individual types. Kleinberg and Raghu [15] consider the use of individual scores to estimate team scores and use these to approximately determine the best team from a pool of agents. Singla et al. [21] present algorithms for learning individual types to form a single team under an online budgeted learning setting. These works concentrate on pure learning. In contrast, our focus is on minimizing regret. Finally, there is a line of work on strategic behavior in teams, studying how to incentivize workers to exert effort [2,3], and how to use signaling to influence team formation [11]. While our work eschews strategic considerations, it suggests extensions that combine learning by the principal with strategic actions by agents.

2 Model

2.1 Agents, Types, and Teams

We consider n *agents* who must be paired by a principal into *teams of two* over a number of rounds; throughout, we assume that n is even. Each agent

has an unknown latent *type* $\theta_i \in \{0, 1\}$. These types can represent any dichotomous attribute: "left-brain" vs. "right-brain" (Sect. 3), "low-skill" vs. "high-skill" (Sects. 4 and 5), etc. We let k denote the number of agents with type 1, and assume that k is fixed *a priori* but unknown.

In each round t, the principal selects a matching M_t, with each edge $(i, j) \in M_t$ representing a *team*. We use the terms "edge" and "team" interchangeably. The *success* of a team $(i, j) \in M_t$ is $f(\theta_i, \theta_j)$, where $f : \{0, 1\}^2 \to \mathbb{R}$ is some known symmetric function of the agents' types. In Sects. 3-5, we restrict our focus to Boolean functions, interpreting $f(\theta_i, \theta_j) = 1$ as a success and $f(\theta_i, \theta_j) = 0$ as a failure. The algorithm observes the success of each team, and may use this to select the matchings in subsequent rounds; however, the algorithm cannot directly observe agents' types. For any matching M, we define its *score* as $S(M) := \sum_{(i,j)\in M} f(\theta_i, \theta_j)$—in the special case of Boolean functions, this is the number of successful teams.

A convenient way to view the Boolean setting is as constructing an edge-labeled *exploration graph* $G(V, E_1, E_2, \ldots)$, where nodes in V are agents, and the edge set $E_t := \bigcup_{t' \le t} M_{t'}$ represents all pairings played up to round t. Upon being played for the first time, an edge is assigned a label $\{0, 1\}$ corresponding to the success value of its team. *Known 0-agents* and *known 1-agents* are those whose types can be inferred from the edge labels. The remaining agents are *unknown*. The *unresolved subgraph* is the induced subgraph on the unknown agents.

2.2 Adversarial Types and Regret

The principal makes decisions facing an *adaptive adversary*, who knows k (unlike the principal, who only knows n) and, in each round, is free to assign agent types *after seeing the matching chosen by the principal*, as long as (1) the assignment is consistent with prior observations (i.e., with the exploration graph), and (2) the number of 1-agents is k. Note that this is the strongest notion of an adversary we can consider in this setting; in particular, since the adversary is fully adaptive and knows the matching *before* making decisions, randomizing does not help, and so it is without loss of generality to consider only deterministic algorithms.

We evaluate the performance of algorithms in terms of additive *regret* against such an adversary. Formally, let M^* be any matching maximizing $S(M^*)$—note that for any Boolean team success function, $S(M^*)$ is a fixed function of n and k. In round t, an algorithm incurs regret $r_t := S(M^*) - S(M_t)$, and its total regret is the sum of its per-round regret over an a priori infinite time horizon. Note, however, that after a finite number of rounds, a naïve algorithm that enumerates all matchings can determine, and henceforth play, M^*; thus, the optimal regret is always finite. Moreover, the "effective" horizon (i.e., the time until the algorithm learns M^*) of our algorithms is small.

2.3 Symmetry Synergy Functions and Atomic Primitives

In subsequent sections, we consider the problem of minimizing regret under four Boolean synergy functions $f\colon \{0,1\}^2 \to \{0,1\}$: EQ, XOR, OR, and AND. Interestingly, the algorithms for these four settings suffice to handle any symmetric synergy function $f\colon \{0,1\}^2 \to \mathbb{R}$. We argue this below for synergy functions that take at most two values; We handle the case of synergy functions f taking three different values at the end of Sect. 3.1.

Lemma 1. *Fix some $\ell \leq u$, let $f\colon \{0,1\}^2 \to \{\ell, u\}$ be any symmetric synergy function, and let $r^f(n,k)$ denote the optimal regret with n agents, of which k have type* 1.

Then, $r^f(n,k) = (u-\ell) \cdot r^g(n,k)$ for one of $g \in \{$EQ,XOR,AND,OR$\}$.

Proof. First, note that without loss of generality, we may assume that $f(0,0) \leq f(1,1)$. Otherwise, we can swap the labels of the agent types without altering the problem. Note that this immediately allows us to reduce team formation under the Boolean NAND and NOR function to the same problem under AND and OR, respectively. Next, note that if $f(0,0) = f(1,0) = f(1,1)$, then the problem is trivial, as all matchings have the same score. Otherwise, we may apply the affine transformation $f \mapsto \frac{1}{u-\ell} \cdot f - \frac{\ell}{u-\ell}$ to the output to recover a Boolean function:

- When $f(0,1) < f(0,0) = f(1,1)$, we recover the EQ function.
- When $f(0,0) = f(1,1) < f(0,1)$, we recover the XOR function.
- When $f(0,0) = f(0,1) < f(1,1)$, we recover the AND function.
- When $f(0,0) < f(0,1) = f(1,1)$, we recover the OR function.

The structure of the problem remains unchanged since total regret is linear in the number of each type of team played over the course of the algorithm. The regret simply scales by a factor of $u - \ell$. ■

3 Uniform and Diverse Teams

We first focus on forming teams that promote uniformity (captured by the Boolean EQ function) or diversity (captured by the XOR function). In addition, we also show that the algorithm for EQ minimizes regret under any general symmetric synergy function taking three different values.

3.1 Uniformity (EQ)

We first consider the *equality* (or EQ) synergy function, $f^{\mathsf{EQ}}(\theta_i, \theta_j) = \overline{\theta_i \oplus \theta_j}$. Here, an optimal matching M^* includes as few $(0,1)$-teams as possible, and thus $S(M^*) = \frac{n}{2} - (k \bmod 2)$. If k (and thus $n-k$) is even, then all agents can be paired in successful teams; else, any optimal matching must include one unsuccessful team with different types. For this setting, Theorem 1 shows that a simple policy (Algorithm 1) achieves optimal regret for *all* parameters n and k.

Algorithm 1. Form Uniform Teams

Round 1: Play an arbitrary matching.
Round 2: Swap unsuccessful teams in pairs as $\{(a,b),\ (c,d)\} \rightarrow \{(a,c),(b,d)\}$. Repeat remaining teams (including one unsuccessful team when k is odd).
Round 3: If $\{(a,b),(c,d)\}$, $\{(a,c),(b,d)\}$ are both unsuccessful, play $\{(a,d),(b,c)\}$. Repeat remaining teams.

Theorem 1. *Define* $r^{\mathsf{EQ}}(n,k) := 2 \cdot \big(\min(k, n-k) - (k \bmod 2)\big)$. *Then,*

1. *Algorithm 1 learns an optimal matching by round 3, and incurs regret at most* $r^{\mathsf{EQ}}(n,k)$.
2. *Any algorithm incurs regret at least* $r^{\mathsf{EQ}}(n,k)$ *in the worst case.*

Proof. For the upper bound on the regret, note that every unsuccessful team includes a 0-agent and a 1-agent. Thus, there is a re-pairing of any two unsuccessful teams that gives rise to two successful teams. If the re-pairing in round 2 is unsuccessful, the only other re-pairing, selected in round 3, must be successful. There will be k mod 2 unsuccessful teams in round 3, making it an optimal matching. At most $\min(k, n-k)$ $(0,1)$-teams can be chosen in each of rounds 1–2, implying that the maximum regret in each of these rounds is $\min(k, n-k) - (k \bmod 2)$. Since Algorithm 1 incurs regret only in rounds 1–2, its total regret is at most $r^{\mathsf{EQ}}(n,k)$.

For the converse (Claim 2), we argue that against *any* algorithm, the adversary can always induce regret $\min(k, n-k) - (k \bmod 2)$ in each of rounds 1–2. Note that after round 2, the exploration graph is a union of two (not necessarily disjoint) matchings, and hence consists of a disjoint union of even-length cycles and isolated (duplicated) edges; this is independent of the algorithm, as it holds for any pair of perfect matchings. Since the graph is bipartite, the adversary can assign types such that no pair of the minority type is adjacent in the graph by starting with the labeling according to the bipartition, then arbitrarily relabeling a subset of the minority side to make the labeling consistent with k. ■

A similar argument allows us to complete our treatment of general (symmetric) synergy functions from Sect. 2.3.

Corollary 1. *For any symmetric synergy function* $f : \{0,1\}^2 \rightarrow \mathbb{R}$ *such that* $f(0,0) \neq f(0,1) \neq f(1,1)$, *there is a regret-minimizing algorithm that locates an optimal matching within two rounds.*

Proof. By applying an affine transformation to the outputs as in Sect. 2.3, we may assume without loss of generality that $f(0,0) = 0$, and $f(1,1) = 1$. There are three cases to consider:

- $f(0,1) = \frac{1}{2}$: The problem is trivial, since all matchings have the same score.
- $f(0,1) > \frac{1}{2}$: The optimal matching includes as many 1–0 agent teams as possible. After the first (arbitrary) matching, every agent is either part of a known 1–0 team or has a known identity (as a member of a 0–0 or 1–1 team). Thus, one can always select an optimal matching in the second round.

– $f(0,1) < \frac{1}{2}$: The optimal matching includes as many 1–1 agent teams as possible, just as in the EQ setting. Note that the three distinct values of f allow us to distinguish between $(0,0)$, $(0,1)$, and $(1,1)$ teams. The same adversarial policy ensures that all 0–1 teams remain sub-optimally paired in round 2, so we exactly recover the EQ setting.

■

3.2 Diversity (XOR)

Next we consider the XOR success function, $f^{\mathsf{XOR}}(\theta_i, \theta_j) = \theta_i \oplus \theta_j$, which promotes diverse teams. Now $S(M^*) = \min(k, n-k)$, since any optimal matching M^* includes as many $(0,1)$-teams as possible. Define $x^+ := \max(0, x)$; we again show that a simple policy (Algorithm 2) has optimal regret for all n, k.

Algorithm 2. Form Diverse Teams

Round 1: Play an arbitrary matching; let $\{(1,2), \ldots, (\ell-1, \ell)\}$ denote unsuccessful teams.
Round 2: Replay all successful teams, and construct a single cycle over all unsuccessful teams (i.e., play teams $\{(\ell, 1), (2,3), \ldots, (\ell-2, \ell-1)\}$).
Round 3: Play any inferred optimal matching (see Theorem 2).

Theorem 2. *Define* $r^{\mathsf{XOR}}(n,k) := 2 \cdot \big(\min(k, n-k) - 1 - (k \bmod 2)\big)^+$. *Then,*

1. *Algorithm 2 learns an optimal matching after round 2, and incurs regret at most* $r^{\mathsf{XOR}}(n,k)$.
2. *Any algorithm incurs regret at least* $r^{\mathsf{XOR}}(n,k)$ *in the worst case.*

Proof. For the achievability in Claim 1, note that each edge $\big(i, (i+1) \bmod \ell\big)$ of the cycle constructed in the algorithm has the following property: if the edge is successful in round 2, then its endpoints have opposite types; otherwise, they have the same type. By following edges around the cycle, the algorithm can therefore construct the sets $S^{=}$ of agents with the same type as agent 1, and $S^{\neq}$ of agents with the opposite type. Subsequently, it is optimal to match $\min(|S^{=}|, |S^{\neq}|)$ teams of (known) opposite-type agents, and match the extraneous agents into unsuccessful teams.

Among agents $1, \ldots, \ell$, there are $k - \frac{n-\ell}{2}$ 1-agents and $n - k - \frac{n-\ell}{2}$ 0-agents; thus, the round 1 regret is $r_1 := \min(k, n-k) - \frac{n-\ell}{2}$ (note that When k is odd, one team must be successful in round 1). Since no regret is incurred after round 2, the adversary must maximize the regret in round 2 conditioned on the choice of ℓ. This is achieved by assigning type 1 to agents $1, \ldots, k - \frac{n-\ell}{2}$, and type 0 to agents $k - \frac{n-\ell}{2} + 1, \ldots, \ell$. Since $(\ell, 1)$ and $(k - \frac{n-\ell}{2}, k - \frac{n-\ell}{2} + 1)$ are the only successful teams (as long as agents 1 to ℓ do not all have the same type), the

regret in round 2 is $(r_1 - 2)^+$. The total regret $\big(2\min(k, n-k) - n + \ell - 2\big)^+$ is monotone increasing in ℓ, with the maximum attained at $\ell = n - 2(k \bmod 2)$. Substituting, we get the upper bound.

For the converse (Claim 2), we describe a policy for the adversary that ensures regret at least $r^{\mathsf{XOR}}(n, k)$. In round 1, the adversary reveals $k \bmod 2$ successful teams, resulting in regret $\min(k, n-k) - (k \bmod 2)$. In round 2, the exploration graph must consist of a disjoint union of even-length cycles (including isolated duplicated edges).

First, when k is odd, consider the component containing the one revealed successful team from round 1. If the component has just two agents (i.e., the algorithm repeats the team), then we again get one successful team. Otherwise, if the team is part of a longer cycle, the adversary puts an odd number of adjacent 0s and an odd number of adjacent 1s in the cycle, such that the previously successful team is (0, 1). Since the edge is not repeated, and only one other (0, 1)-team is created, the algorithm gets at most one successful team in this cycle. The remaining cycles contain an even number of 1-agents, so we appeal to below.

When k is even, the adversary fills cycles with 0-agents until they are exhausted, then labels all remaining agents as 1-agents. At most one cycle contains both agent types. Placing the 0-agents contiguously in this cycle ensures only two adjacent successful teams. Since all cycle lengths are even, as is $n - k$, these successful teams will be an even number of edges apart; in particular, the adversary can ensure that they are both edges from round 2, making the assignment consistent with round 1. In total, the algorithm obtains at most $2 + (k \bmod 2)$ successful teams in round 2, giving total regret at least $r^{\mathsf{XOR}}(n, k)$. ■

4 The Strongest Link Setting (OR)

We next consider the Boolean OR synergy function, that is, $f^{\mathsf{OR}}(\theta_i, \theta_j) = \theta_i + \theta_j$. Adopting the terminology of Johari et al. [12], we refer to this setting as the *strongest link* model: interpreting 0/1-agents as having low/high skill, a team is successful when it has at least one high-skill member.

Observe that under OR, we have $S(M^*) = \min(k, n/2)$, since any optimal matching M^* includes a maximal set of $(0, 1)$-teams. Define $\alpha := \frac{n-k}{n}$ to be the *fraction of low-skill agents*; our regret bounds in this setting are more conveniently phrased in terms of α. In particular, our first result establishes the following lower bounds on the regret incurred by *any* algorithm.

Theorem 3. *For the strongest link setting, any algorithm incurs regret at least* $L^{\mathsf{OR}}(\alpha) \cdot n$ *in the worst-case, where*

$$L^{\mathsf{OR}}(\alpha) = \begin{cases} \frac{13\alpha}{17} & 0 \le \alpha \le \frac{1}{2} \\ \frac{6-9\alpha}{4} & \frac{1}{2} < \alpha \le \frac{6}{11} \\ \frac{3-4\alpha}{3} & \frac{6}{11} < \alpha \le \frac{3}{5} \\ \frac{1-\alpha}{2} & \frac{3}{5} < \alpha \le 1 \end{cases}$$

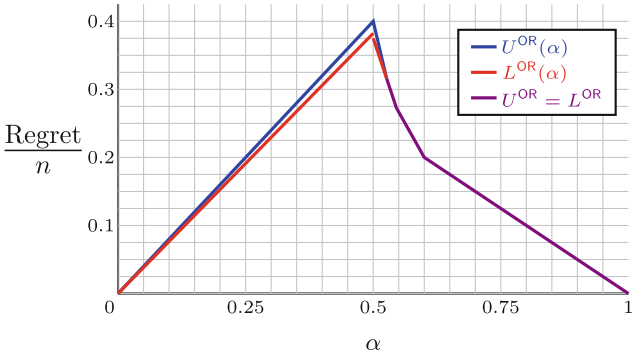

Fig. 1. *Our regret bounds (Thoerems 3 to 5) under the Strongest Link model, as functions of* $\alpha := \frac{n-k}{n}$*, the fraction of low-skill agents. The bounds match for* $\frac{10}{19} \leq \alpha \leq 1$.

Proof Sketch. The bounds are established using a common underlying adversarial strategy that forces any algorithm to incur an unavoidable regret. The structure of the strategy is as follows: The adversary first reveals a chosen fraction of 0-agents in round 1. Subsequently, when the algorithm explores an unresolved agent, the adversary reveals it to be a 0-agent whenever possible. This leads to a tension between inducing high regret in the first round (by revealing 0-edges), and leaving more unresolved 0-agents for later rounds. $L^{\mathsf{OR}}(\alpha)$ is obtained by choosing the fraction of initially revealed 0-agents to maximize regret under this tension. A full proof is presented in the extended version. ■

The lower bound in Theorem 3 is plotted in Fig. 1, and notably varies greatly with α. Nevertheless, we provide a policy (Algorithm 3) that manages to achieve *nearly matching regret across all* α, while being agnostic of k (and thus α). Both bounds are plotted in Fig. 1; despite the functions being piecewise linear, they match exactly for $\alpha \geq \frac{10}{19}$, and $U^{\mathsf{OR}}(\alpha) - L^{\mathsf{OR}}(\alpha) < 0.018$ for all α.

4.1 The MaxExploit with 4-Cliques Algorithm

To simplify our analysis, we introduce some terminology: we say that two unknown 0-agents become *discovered* when they are paired to form an unsuccessful team. An unknown agent is *explored* when its type is revealed by pairing it with a known 0-agent. Our policy for this setting, MaxExploit with 4-Cliques, is given in Algorithm 3. The algorithm exploits the inferred types of agents to the greatest possible extent; a maximal number of known 1–0 agent teams are played in each round. Exploration is only done using known 0-agents that cannot be included in such a pair. If only two agents in a 4-cycle are explored, we treat the other two agents as unknown, even if their type is deducible.

First, to see that the algorithm terminates, note that in each iteration of the loop, at least one known 0-agent is used for exploration, revealing the type of another agent. Thus, the algorithm makes progress and eventually terminates.

Algorithm 3. MaxExploit with 4-Cliques

Round 1: Select an arbitrary matching.
while #{known 0-agents} > #{known 1-agents} **and** #{unknown agents} > 0 **do**
Pair each known 1-agent with a known 0-agent.
Use extra known 0-agents to explore both members of successful teams.
(In round 3, explore all members of 4-cycles whenever possible[3].)
Round 2: Re-pair remaining unknown successful teams into 4-cycles.
(If number of remaining unknown successful teams is odd, repeat one team.)
Round 3: In each 4-cycle with undiscovered agents, re-pair to form a 4-clique.
Round 4+: Re-play the matching from round 1 on unexplored successful teams.

Next, note that unknown agents are always in successful teams throughout the algorithm (as both members of an unsuccessful team can be deduced as 0-agents). Upon termination, the algorithm can play an optimal matching: either all agents are known, or there are enough known 1-agents to match all known 0-agents, and the other successful teams of unknown agents can be safely replayed.

Let d_t be the number of 0-agents discovered in round t by pairing two unknown agents, and e_t the number of 0-agents revealed by exploration with a known 0-agent. We define

$$\Delta_t := \#\{\text{known 0-agents after round } t\} - \#\{\text{known 1-agents after round } t\}$$

The following lemma studies how Δ_t evolves over rounds t.

Lemma 2. *$\Delta_1 = d_1$, and $2e_t = \Delta_t \leq \Delta_{t-1}$, for all $t \geq 2$.*

Proof. In round 1, the algorithm discovers d_1 0-agents, and no 1-agent (since there is no exploration); hence $\Delta_1 = d_1$. Consider the 4-cycle and 4-clique edges played in rounds 2–3. If such an edge comprises two 0-agents, then the other two agents in its cycle or clique must be 1-agents. In particular, the addition of d_t known 0-agents in these rounds is exactly counterbalanced by the deduction of their neighboring d_t 1-agents, so discovery does not contribute to $\Delta_{t+1} - \Delta_t$.

Next, consider any round $t \geq 2$. The algorithm first pairs all known 1-agents with known 0-agents, so exactly Δ_{t-1} agents are used for exploration. Each exploration must discover either a 0-agent or a 1-agent, so $\Delta_t = \Delta_{t-1} + e_t - (\Delta_{t-1} - e_t) = 2e_t$. Since members of successful teams are explored in pairs, at most half of all explorations can reveal 0-agents. Thus, $e_t \leq \frac{\Delta_{t-1}}{2}$. ■

For the subsequent analysis, there are two distinct regimes depending on the *fraction* of low-skill agents α. When most agents are low-skill ($\alpha > \frac{1}{2}$), the optimal configuration includes some $(0,0)$-teams, but no $(1,1)$-teams, and r_t equals the number of $(1,1)$-teams in M_t. On the other hand, when most agents are high-skill ($\alpha \leq \frac{1}{2}$), the optimal configuration consists entirely of successful teams, and an algorithm's round-t regret r_t is the number of $(0,0)$-teams in M_t. Consequently, the analysis in each regime is very different.

4.2 Majority High-Skill Regime ($\alpha \leq \frac{1}{2}$)

We begin the analysis by focusing on the case when $\alpha \leq \frac{1}{2}$. Recall that the total regret in this regime equals the total number of $(0,0)$ teams the algorithm plays.

Theorem 4. *For $\alpha \leq \frac{1}{2}$, Algorithm 3 has regret at most $\frac{4}{5} \cdot \alpha n$.*

Proof. First, note that in the regime $\alpha \leq \frac{1}{2}$, the algorithm never pairs two *known* 0-agents; known 0-agents are paired with known 1-agents or used for exploration. Hence, the number of $(0,0)$-teams selected, and thus the regret, in round t is $e_t + \frac{d_t}{2}$. (Note that $e_1 = 0$.)

After round 3, by Lemma 2, there are $2e_3$ more known 0-agents than 1-agents. The unresolved agents are contained in 4-cliques of successful teams, which must each contain at least three 1-agents. Thus, exploring any 0-agent means that the algorithm can deduce three 1-agents. After e_3 such explorations, the algorithm locates $3e_3$ 1-agents, terminating the loop. The regret incurred in rounds 4 and later is thus at most e_3, giving total regret at most $\frac{d_1}{2} + \frac{d_2}{2} + \frac{d_3}{2} + e_2 + 2e_3$. We can now bound the regret incurred by Algorithm 3 by formulating the adversary's problem of choosing the worst-case number of revealed zeros in each round as an LP with variables $\{d_1, d_2, d_3, e_2, e_3\}$. Applying Lemma 2 to rounds 2 and 3, we obtain that $e_2 \leq \frac{d_1}{2}$ and $e_3 \leq e_2$. In addition, $d_1 + d_2 + d_3 + e_2 + 2e_3 \leq \alpha n$ ensures that the number of 0-agents revealed by the adversary is at most the total number of 0-agents. Put together, we get the following LP:

$$\begin{aligned} \text{Maximize:} \quad & \tfrac{d_1}{2} + \tfrac{d_2}{2} + \tfrac{d_3}{2} + e_2 + 2e_3 \\ \text{Subject to:} \quad & e_2 \leq \tfrac{d_1}{2} \\ & e_3 \leq e_2 \\ & d_1 + d_2 + d_3 + e_2 + 2e_3 \leq \alpha n \\ & d_1, d_2, d_3, e_2, e_3 \geq 0 \end{aligned}$$

Solving, we get $(d_1, d_2, d_3, e_2, e_3) = (\frac{2\alpha n}{5}, 0, 0, \frac{\alpha n}{5}, \frac{\alpha n}{5})$ as the adversary's best strategy, with regret at most $\frac{4}{5}\alpha n$. ∎

4.3 Majority Low-Skill Regime ($\alpha > \frac{1}{2}$)

A different, more involved, analysis shows that Algorithm 3 is also near-optimal when $\alpha > \frac{1}{2}$.

Theorem 5. *For $\alpha > \frac{1}{2}$, Algorithm 3 learns an optimal matching after incurring regret at most $U^{OR}(\alpha) \cdot n$, where*

$$U^{OR}(\alpha) = \begin{cases} \frac{10-16\alpha}{5} & \frac{1}{2} \leq \alpha < \frac{10}{19}, \\ \frac{6-9\alpha}{4} & \frac{10}{19} \leq \alpha < \frac{6}{11}, \\ \frac{3-4\alpha}{3} & \frac{6}{11} \leq \alpha < \frac{3}{5}, \\ \frac{1-\alpha}{2} & \frac{3}{5} \leq \alpha \leq 1. \end{cases}$$

Note that $\lim_{\alpha \downarrow \frac{1}{2}} U^{\mathsf{OR}}(\alpha) = \frac{2}{5}$, which matches $\lim_{\alpha \uparrow \frac{1}{2}} U^{\mathsf{OR}}(\alpha)$ from Theorem 4. Before proceeding, we define $s_t^{00}, s_t^{01}, s_t^{11}$ to be the number of $(0,0)$, $(0,1)$, and $(1,1)$-teams the algorithm plays in round t, respectively. Since there are $(1-\alpha)n$ 1-agents in total, $s_t^{01} = (1-\alpha)n - 2s_t^{11}$; in turn, since there are αn 0-agents, $s_t^{00} = \frac{1}{2} \cdot (\alpha n - s_t^{01}) = s_t^{11} + (\alpha - \frac{1}{2}) \cdot n > s_t^{11}$. We now prove Theorem 5 via a series of lemmas.

Lemma 3. *The adversary has a best response to Algorithm 3 with the following properties:*

1. *It never reveals pairs of unknown agents as* $(0,0)$*-teams after round 1.*
2. *It never reveals any* $(1,1)$*-team until all* $(0,1)$*-teams have been revealed.*

Proof. For the first claim, suppose that the adversary reveals a $(0,0)$-team among the re-paired teams in round $t = 2$ or $t = 3$. The 4-cycle or 4-clique containing this $(0,0)$-team contains two 0-agents and two 1-agents, so two $(0,1)$-teams were selected in each of the first $t-1$ rounds. The adversary can force the same regret, and provide the same information, by relabeling these agents so a $(0,0)$-team and a $(1,1)$-team are revealed in round 1, the two 1-agents are explored in round 2, and (if $t = 3$) the $(0,1)$-teams are repeated in round 3. By repeating this relabeling, we arrive at an adversary strategy of the same regret, of the claimed form.

For the second claim, recall that the algorithm's regret in the regime $\alpha > \frac{1}{2}$ is exactly the number of $(1,1)$-teams it plays. We will describe a scheme charging $(1,1)$-teams played in round t to 0-agents explored in round t.

Consider some round $t \geq 2$ in which s_t^{11} $(1,1)$-teams are played. Since $s_t^{00} > s_t^{11}$, and all $(0,0)$-teams result from exploration[1] by the first claim, we can charge one distinct explored 0-agent for each such $(1,1)$-team. Thus, the number of *uncharged* explored 0-agents in round t is exactly $s_t^{00} - s_t^{11} = (\alpha - \frac{1}{2}) \cdot n$, independent of t and s_t^{11}.

The total number of 0-agents explored in rounds $t \geq 2$ is exactly s_1^{01}. If the algorithm runs for T rounds, exactly $(T-1) \cdot (\alpha - \frac{1}{2}) \cdot n$ explored 0-agents remain uncharged. Thus, the number of *charged* 0-agents, which equals the regret incurred after round 1, is $s_1^{01} - (T-1) \cdot (\alpha - \frac{1}{2}) \cdot n$. Conditioned on $s_1^{00}, s_1^{01}, s_1^{11}$, the regret is therefore maximized by minimizing T; that is, the adversary wants the algorithm to finish in as few rounds as possible. To minimize the number of rounds T, the adversary should maximize the number of 0-agents available for exploration. The adversary accomplishes this by having the algorithm explore $(0,1)$-teams before any $(1,1)$-teams. ■

We now focus, without loss of generality, on such an adversary. This lets us bound the regret in terms of $(s_1^{00}, s_1^{01}, s_1^{11})$.

[1] Except in the last round, where known (0, 0)-teams may be played; however, no (1, 1)-teams are played in this round.

Lemma 4. *Conditioned on $s_1^{00}, s_1^{01}, s_1^{11}$, Algorithm 3 has regret at most*

$$\left\lfloor \frac{s_1^{01}+s_1^{11}}{s_1^{00}} \right\rfloor s_1^{11} + \min(s_1^{11}, (s_1^{01}+s_1^{11}) \mod s_1^{00}).$$

Proof. From Lemma 3 we know that only $(0,1)$-teams are explored before any $(1,1)$-team is explored. Each explored $(0,1)$-team results in pairing a 0-agent with a newly discovered 1-agent, and also adds a known 0-agent. Thus as long as the algorithm explores only $(0,1)$-teams, the number of pairs of 0-agents available for exploration stays constant at s_1^{00}. Therefore, the total number of rounds of exploration until all agents in successful teams are explored is $\left\lceil \frac{s_1^{01}+s_1^{11}}{s_1^{00}} \right\rceil$. One subtlety here is that the exploration of $(1,1)$-teams decreases the available 0-agents. However, because $s_1^{11} < s_1^{00}$, the first round in which a $(1,1)$-team can be explored is one round before the last; in this case, the last round only explores $(1,1)$-teams, and the number of 0-agents available for this exploration exceeds the number of 1-agents to be explored. Thus, the bound on the number of rounds of exploration does hold. In the last round of exploration, no $(1,1)$-teams can be played, so no regret is incurred. We therefore focus on the first $T = \left\lfloor \frac{s_1^{01}+s_1^{11}}{s_1^{00}} \right\rfloor$ rounds of exploration. Again because $s_1^{11} < s_1^{00}$, none of the first $T-1$ rounds of exploration explore any $(1,1)$-team; thus, each of these rounds, as well as the very first round of the algorithm, incurs a regret of s_1^{11}.

In round T, the total regret is the number of $(1,1)$-teams explored in round $T+1$ (because these teams are still played in round T). This number is either $(s_1^{01}+s_1^{11}) \mod s_1^{00}$ (if *only* $(1,1)$-teams are explored in round $T+1$, then it is the total number of explored teams in round $T+1$), or s_1^{11} (if some $(0,1)$-teams are explored in round $T+1$, then *all* $(1,1)$-teams are explored in round $T+1$). Thus, the regret in the T^{th} round of exploration is the minimum of the two terms. We thus obtain the total regret of the algorithm as: $\left\lfloor \frac{s_1^{01}+s_1^{11}}{s_1^{00}} \right\rfloor \cdot s_1^{11} + \min(s_1^{11}, (s_1^{01}+s_1^{11}) \mod s_1^{00})$. ∎

s_t^{00} turns out to be further constrained, as follows:

Lemma 5. *In Algorithm 3, if the adversary reveals 0-agents only by exploration in rounds $t \geq 2$, then $s_1^{00} > \frac{\alpha}{5} n$.*

Proof. The number of 0-agents discovered in round 1 is $2s_1^{00}$. In rounds 2 and 3 combined, the algorithm discovers an additional $e_2 + e_3$ 0-agents. The number of 1-agents discovered in round 2 is $\Delta_1 - e_2 = 2s_1^{00} - e_2$, and in round 3, it is $\Delta_2 - e_3 = 2e_2 - e_3$, by Lemma 2. Thus, the number of unknown 0-agents after round 3 is $\alpha n - 2s_1^{00} - e_2 - e_3$, and the number of unknown 1-agents is $(1-\alpha)\cdot n - 2s_1^{00} - e_2 + e_3$. But notice also that after round 3, all unknown agents form 4-cliques of successful edges, which can contain at most one 0-agent each. Therefore, there must be at least three times as many remaining 1-agents as 0-agents, so $(1-\alpha)\cdot n - 2s_1^{00} - e_2 + e_3 \geq 3(\alpha n - 2s_1^{00} - e_2 - e_3)$. Rearranging, we obtain that $4s_1^{00} \geq (4\alpha - 1)\cdot n - 2e_2 - 4e_3$. By Lemma 2, we get that $e_3 \leq e_2 \leq s_1^{00}$, so the previous inequality in particular implies that $4s_1^{00} \geq (4\alpha - 1)\cdot n - 6s_1^{00}$, or $s_1^{00} \geq \frac{4\alpha-1}{10} \cdot n > \frac{\alpha}{5} \cdot n$, because $\alpha > \frac{1}{2}$. ∎

We obtain the piecewise-linear bound in Theorem 5 by maximizing the bound in Lemma 4 subject to the constraint in Lemma 5 (and using $s_t^{01} = \alpha n - 2s_t^{00}$, $s_t^{11} = s_t^{00} - (\alpha - \frac{1}{2})n$). Details of this calculation can be found in the extended version.

5 The Weakest Link Setting (AND)

Finally, we consider the Boolean AND synergy function. If, as before, we interpret 0/1-agents as having low/high skill, then (in the terminology of Johari et al. [12]), this corresponds to a *weakest link* model: the difficulty of the task ensures any team with a low-skill member is unsuccessful. To simplify the analysis, we assume throughout that k is even.

Theorem 6. *For the weakest link model, any algorithm incurs regret at least* $L^{AND}(n,k) := n - k$.

Proof Sketch. The total regret for an algorithm equals half the number of $(0,1)$-teams selected over the duration of the algorithm, since the optimal solution would re-pair these agents into $(0,0)$-teams and $(1,1)$-teams. We consider a myopic greedy adversary which reveals as few 1-agents as possible in each round, and argue that such an adversary can ensure that each 0-agent is paired with at least two 1-agents during the execution of any algorithm. The proof is presented via a series of lemmas in the extended version. ■

5.1 The Ring Factorization with Repairs Algorithm

The fact that the regret of an algorithm is half the number of $(0,1)$-teams selected suggests that we want the algorithm's chosen matchings to quickly locate (and pair) all of the 1-agents, while minimizing the number of times each 0-agent is paired with a 1-agent. Playing matchings according to a *1-factorization* (that is, a partition of the complete graph K_n into perfect matchings) ensures that no team is ever repeated. This intuition is used in the Exponential Cliques algorithms of Johari et al. [12], who show that when each agent has independent Bernoulli(k/n) type, this algorithm has expected regret $\frac{3}{4}(n-k) + o(n)$, which is asymptotically optimal. Against an adaptive adversary, however, an arbitrary 1-factorization is not enough to get good regret; for example, a 1-factorization that first builds the Turán graph $T(n, \frac{n}{k})$ [1,22] has regret $\frac{1}{2}k(n-k)$. Similarly, the performance of Exponential Cliques in the worst case is also much worse.

Lemma 6. Exponential Cliques *incurs regret* $2(n-k-1)$ *against an adaptive adversary.*

Proof. Consider an instance on $n = 2^j + 2$ agents with $k = 2$ having high skill. An adaptive adversary can ensure that the two 1-agents comprise the last unexplored team. Over its first $2^j - 1$ rounds, Exponential Cliques builds a 2^j-clique in the explored subgraph while repeating the remaining team $2^j - 1$ times.

Subsequently, it must spend 2^j additional rounds exploring all teams comprising a member of this repeated edge and a member of the clique, resulting in regret $2(2^j - 1) = 2(n - k - 1)$. ■

Our main algorithm for this setting, Ring Factorization with Repairs, leverages a particular 1-factorization, which we call the Ring Factorization. We organize the agents into two nested rings and choose matchings so that closer agent pairs under this ring geometry are matched earlier. In the first round, agents in corresponding positions in the rings are matched. Over the next four rounds, matchings are chosen to pair each agent with the four agents in adjacent positions in the rings, and this process repeats at greater distances. The structure and order of the four matchings chosen in each "phase" are critical. A formal description of this 1-factorization is given in the extended version. We visualize the first 5 matchings in the constructions for $n = 10$ and $n = 12$ in Fig. 2.

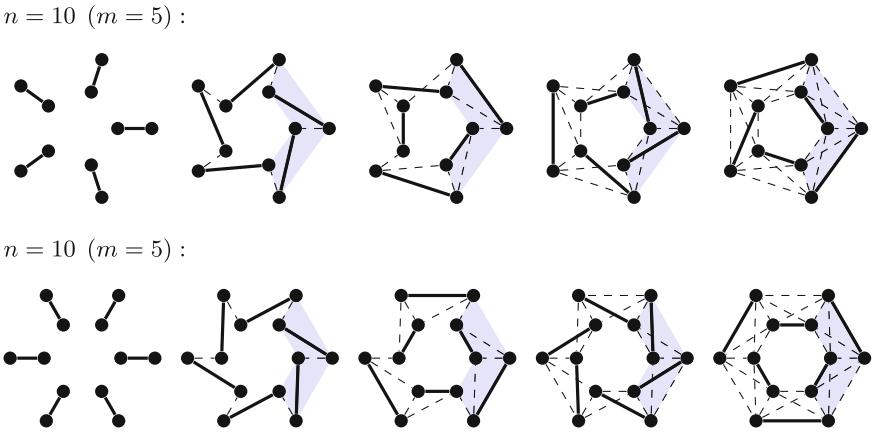

Fig. 2. *The first five rounds (i.e., Phases 0 and 1) of* Ring Factorization *on 10 (top) and 12 (bottom) agents. The last four matchings illustrate the general matching sequence for cycles in intermediate phases; the blue highlighted section of each matching is repeated based upon the size of the cycle.*

Theorem 7. Ring Factorization with Repairs *(Algorithm 4) locates an optimal matching after incurring regret at most* $U^{\mathsf{AND}}(n,k) := n - k + \left\lfloor \frac{\min(k,n-k)}{4} \right\rfloor$.

Proof Sketch. As mentioned before, the double-ring structure of our factorization defines a notion of distance between agents (namely, the difference between their column indices modulo m). By selecting matchings according to this factorization, each agent is paired up with other agents in non-decreasing order of distance. Consider this pairing from the perspective of a 0-agent x. We will show

Algorithm 4. Ring Factorization with Repairs (Sketch)

while #{unknown agents} > 0 **do**
  Select a matching via Ring Factorization for n.
  if a $(1,1)$-team is revealed **then**
    Perform a case-specific "repair" step (possibly over multiple rounds) that partitions agents into known $(0,0)$-teams, known $(1,1)$-teams, and an intermediary stage of the Ring Factorization construction of size $n' < n$. (See Appendix B in the extended version)
    Play known $(0,0)$- and $(1,1)$-teams, and continue playing matchings according to Ring Factorization on the remaining n' agents.

that roughly speaking (with some exceptions which require technical work and slightly weaken the bound), x will be paired with at most two 1-agents before being identified as a 0-agent. If each 0-agent is paired with at most two 1-agents before discovery, then we get an overall regret bound of $n - k$. Consider three 1-agents $\{y_1, y_2, y_3\}$, all located in different columns from x. Then, two of these 1-agents—say, y_1 and y_2—lie on the same side of x. Thus, y_1 and y_2 are strictly closer to each other than the further of the two (say, y_1) is to x. In particular, y_1 and y_2 were paired before y_1 is paired with x, and so must have been revealed as 1-agents. Thus, y_1 will never be paired with x. Since this holds for every triple of 1-agents, x cannot be paired with three 1-agents.

While the above argument encapsulates the main intuition, the technical challenge is removing the assumption that y_1, y_2, y_3 were all in different columns from x. "Repairing" the cycle to account for the case of a 1-agent in the same column as x largely accounts for the additional term in the regret bound. The full analysis of these "repair" steps is intricate; see Appendix B in the extended version for details. ■

The bounds of Theorems 6 and 7 are off by an additive term $\lfloor \min(k, n-k)/4 \rfloor$. The lower bound is simpler, and it is tempting to think that it may be tight; unfortunately, this is not true in general; in Appendix C of the extended version, we show that any algorithm on the instance with $n = 10$, $k = 4$ must incur regret at least 7 (which coincidentally matches the upper bound in Theorem 7, though it is unclear if this extends to larger settings). Closing this gap is an interesting and challenging direction for future work.

6 Conclusion

Our work provides near-optimal regret guarantees for learning an optimal matching among agents under any symmetric function of two binary variables. While our results are specific to each function, they exhibit several noteworthy common features. First, although we consider an adaptive adversary, it is not hard to see that the regret bounds with i.i.d. Bernoulli types can only improve by a small constant factor; such a small gap between stochastic and adversarial models is uncommon. Next, for all our settings, minimizing regret turns out

to require maximal exploitation (in contrast to quickly learning all agent types, which would benefit from more exploration). Finally, the problems appear to get harder for $k = \frac{n}{2}$, and also handling the weakest link setting (i.e., the Boolean AND function) is more challenging than other synergy functions. These phenomena hint at underlying information-theoretic origins, and formalizing these may help in reasoning about more complex models.

Our work raises three natural future directions:

1. It would be desirable to close the gaps between our bounds for the OR and AND settings. In each case, however, our results suggest that the optimal procedures may depend heavily on number-theoretic properties of n and k which can expose a further level of complication.
2. We consider only perfect feedback, which in itself presented interesting challenges, but may be unrealistic in real-world settings. Our results likely extend to some noisy feedback models by repeatedly playing a team and averaging their scores. However, quantifying the relationship between the amount of noise and the expected additional regret is an open problem.
3. The restriction to teams of size 2, and binary agent types, are the main restrictions of our model. For a more general theory of team formation, it is desirable to consider larger teams and other synergy functions (in particular, threshold functions); doing so is a rich and challenging open direction.

Acknowledgments. We would like to thank anonymous reviewers for useful feedback. Part of the work was done when SB and ME were visiting the Simons Institute for the Theory of Computing for the semester on Data-Driven Decision Processes; they also acknowledge support from the NSF under grants ECCS-1847393 and CNS-195599, and the ARO MURI grant W911NF1910217. DK acknowledges support from ARO MURI grant ARO W911NF1810208.

References

1. Aigner, M.: Turán's graph theorem. Am. Math. Mon. **102**(9), 808–816 (1995)
2. Babaioff, M., Feldman, M., Nisan, N.: Combinatorial agency. In: Proceedings of 7th ACM Conference on Electronic Commerce, pp. 18–28 (2006)
3. Carlier, G., Ekeland, I.: Matching for teams. Econ. Theory **42**, 397–418 (2010)
4. Cesa-Bianchi, N., Lugosi, G.: Combinatorial bandits. J. Comput. Syst. Sci. **78**(5), 1404–1422 (2012)
5. Chen, W., Wang, Y., Yuan, Y.: General framework and applications. In: ICML, Combinatorial Multi-armed Bandit (2013)
6. Combes, R., Sadegh Talebi, M., Proutiere, A., Lelarge, M.: Combinatorial bandits revisited. In: Proceedings of 29th Advances in Neural Information Processing Systems, pp. 2116–2124 (2015)
7. Devanur, N.R., Jain, K.: Online matching with concave returns. In: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, pp. 137–144 (2012)
8. Gai, Y., Krishnamachari, B., Jain, R.: Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations. IEEE/ACM Trans. Networking **20**(5), 1466–1478 (2012)

9. Gopalan, A., Mannor, S., Mansour, Y.: Thompson sampling for complex online problems. In: International Conference on Machine Learning, pp. 100–108. PMLR (2014)
10. Han, Y., Wang, Y., Chen, X.: Adversarial combinatorial bandits with general non-linear reward functions. In: International Conference on Machine Learning, pp. 4030–4039 (2021)
11. Hssaine, C., Banerjee, S.: Information signal design for incentivizing team formation. In: Proceedings of 14th Conference on Web and Internet Economics (WINE) (2018)
12. Johari, R., Kamble, V., Krishnaswamy, A.K., Li, H.: Exploration vs. exploitation in team formation. In: Proceedings of 14th Conference on Web and Internet Economics (WINE) (2018)
13. Katariya, S., Kveton, B., Szepesvari, C., Vernade, C., Wen, Z.: Stochastic rank-1 bandits. In: Artificial Intelligence and Statistics, pp. 392–401. PMLR (2017)
14. Kirschner, J., Lattimore, T., Krause, A.: Information directed sampling for linear partial monitoring. In: Conference on Learning Theory, pp. 2328–2369. PMLR (2020)
15. Kleinberg, J., Raghu, M.: Team performance with test scores. In: Proceedings of 16th ACM Conference on Economics and Computation (2015)
16. Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvári. Tight regret bounds for stochastic combinatorial semi-bandits. In Proc. 18th Intl. Conf. on Artificial Intelligence and Statistics, 2015
17. Merlis, N., Mannor, S.: Tight lower bounds for combinatorial multi-armed bandits. In: Conference on Learning Theory, pp. 2830–2857. PMLR (2020)
18. Rajkumar, A., Mukherjee, K., Tulabandhula, T.: Learning to partition using score based compatibilities. In: Proceedings of 16th International Conference on Autonomous Agents and Multiagent Systems, pp. 574–582 (2017)
19. Russo, D., Van Roy, B.: Learning to optimize via information-directed sampling. Advances in Neural Information Processing Systems, 27 (2014)
20. Sentenac, F., Yi, J., Calauzenes, C., Perchet, V., Vojnovic, M.: Pure exploration and regret minimization in matching bandits. In: ICML21, pp. 9434–9442 (2021)
21. Singla, A., Horvitz, E., Kohli, P., Krause, A.: Learning to hire teams. In: Third AAAI Conference on Human Computation and Crowdsourcing (2015)
22. Turán, P.: Egy gráfelméleti szélsoértékfeladatról (on an extremal problem in graph theory). Mat. Fiz. Lapok **48**(3), 436–453 (1941)
23. Wang, S., Chen, W.: Thompson sampling for combinatorial semi-bandits. In: International Conference on Machine Learning, pp. 5114–5122. PMLR (2018)
24. Zimmert, J., Seldin, Y.: Factored bandits. Advances in Neural Information Processing Systems 31 (2018)