Online Team Formation under Different Synergies

Matthew Eichhorn
Cornell University
Ithaca, NY 14850
meichhorn@cornell.edu

Siddhartha Banerjee
Cornell University
Ithaca, NY 14850
sbanerjee@cornell.edu

David Kempe
University of Southern California
Los Angeles, CA 90089
david.m.kempe@gmail.com

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.

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 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 Section 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 Section 3), OR (in Section 4) and AND (in Section 5). While this may at first appear to be a limited class of synergy functions, in Section 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 Section 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, 5, 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 (Section 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" (Section 3), "low-skill" vs. "high-skill" (Sections 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 Sections 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 Section 3.1.

Lemma 1. *Fix some $\ell \le u$, 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 $f(0,0) \le f(1,1)$. 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 $\{(a,b), (c,d)\} \to \{(a,c), (b,d)\}$. 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 $k \bmod 2$ 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 Section 2.3.

Corollary 1. *For any symmetric synergy function $f : \{0,1\}^2 \to \mathbb{R}$ 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 Section 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.

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.*

Algorithm 2 FORM DIVERSE TEAMS

Round 1: Play an arbitrary matching; let $\{(1,2),\dots,(\ell-1,\ell)\}$ denote unsuccessful teams.

Round 2: Replay all successful teams, and construct a single cycle over all unsuccessful teams (i.e., play teams $\{(\ell,1),(2,3),\dots,(\ell-2,\ell-1)\}$).

Round 3: Play any inferred optimal matching (see Theorem 2).

Proof For the achievability in Claim 1, note that each edge $\big(i,(i+1) \mod \ell\big)$ 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 $1,\dots,\ell$, 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 $1,\dots,k-\frac{n-\ell}{2}$, and type 0 to agents $k-\frac{n-\ell}{2}+1,\dots,\ell$. 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 $\ell = n-2(k \mod 2)$. 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 $k \mod 2$ successful teams, resulting in regret $\min(k,n-k)-(k \mod 2)$. 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 $2+(k \mod 2)$ 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*

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

We establish the bound given in this theorem via a sequence of lemmas that present and analyze an adversarial policy for a particular range of α. Since the adversary knows α, he can choose the worst policy, i.e., achieve the pointwise maximum of the regret of the different policies. We begin with a lemma that allows us to restrict our attention to a particular subfamily of algorithms.

Lemma 2. *There exists an optimal algorithm that never pairs known 0-agents until the types of all agents are known.*

Proof We argue the claim with an exchange argument. Suppose that the algorithm pairs two known 0-agents in a round in which it also plays a team (u, v), where at least one of u, v is unknown. If u, v are both 0-agents, then the algorithm will learn this whether it pairs u and v or explores u and v with the known 0-agents. Both of these actions result in the same regret. If exactly one of u, v is a 0-agent, then the algorithm again accrues no additional regret by exploring u, v as opposed to pairing them. Moreover, the algorithm can only gain more information about the types of u, v through exploration. Finally, if both u and v are 1-agents, then the algorithm incurs one less unit of regret and learns the types of both u and v by exploring them with 0-agents as opposed to pairing them. By repeatedly applying these swaps, we never increase the regret of the algorithm, nor reduce the set of deductions that it can make about node types. When the swapping process finishes, we are left with an algorithm of the claimed type. ■

By this lemma, we can, without loss of generality, restrict our attention to algorithms that use all 0-agents for exploration.

Lemma 3. *There is a policy for the adversary that ensures regret at least $\frac{13\alpha}{17} \cdot n$ for all $\alpha \leq \frac{1}{2}$.*

Proof Recall that for $\alpha \leq \frac{1}{2}$, per-round regret is equal to the number of $(0,0)$-teams that were selected. Consider the following adversarial policy:

Round 1: The algorithm chooses an arbitrary matching. The adversary reveals $\frac{4\alpha}{17} \cdot n$ $(0,0)$-teams, giving regret $\frac{4\alpha}{17} \cdot n$.

Round 2: The algorithm explores with its $\frac{8\alpha}{17} \cdot n$ known 0-agents. It will explore members from $\left(\frac{4\alpha}{17} + \beta\right) \cdot n$ teams from round 1, for some $0 \leq \beta \leq \frac{4\alpha}{17}$. The adversary can label one member from each of these teams as a 0-agent, giving regret $\left(\frac{4\alpha}{17} + \beta\right) \cdot n$.

Round 3–4: To this point, the algorithm has learned the types of $\left(\frac{12\alpha}{17} + \beta\right) \cdot n$ 0-agents and $\left(\frac{4\alpha}{17} + \beta\right) \cdot n$ 1-agents (the partners of 0-agents from successful teams). Therefore, the algorithm again explores with $\frac{8\alpha}{17}n$ known 0-agents in round 3; let S be the set of unknown agents that is being explored, of size $|S| = \frac{8\alpha}{17}n$. Let $I \subseteq S$ be a maximum independent set of S in the unexplored subgraph. Since the unexplored subgraph is bipartite, $|I| \geq \frac{|S|}{2} = \frac{4\alpha}{17} \cdot n$.

There remain $\left(\frac{5\alpha}{17} - \beta\right) \cdot n$ unknown 0-agents. If $|I| \geq \left(\frac{5\alpha}{17} - \beta\right) \cdot n$, then the adversary reveals an arbitrary subset of I of size $\left(\frac{5\alpha}{17} - \beta\right) \cdot n$ as 0-agents. This gives regret $\left(\frac{5\alpha}{17} - \beta\right) \cdot n$ in round 3, and brings the total accumulated regret to $\frac{13\alpha}{17}n$.

Otherwise, the adversary reveals all of I as 0-agents. We write $|I| = \left(\frac{4\alpha}{17} + \gamma\right) \cdot n$, for some $0 \leq \gamma \leq \left(\frac{\alpha}{17} - \beta\right)$. The regret in round 3 is then $\left(\frac{4\alpha}{17} + \gamma\right) \cdot n$.

Since the exploration graph after round 2 is 2-regular, each 0-agent revealed in round 3 can result in the revelation of at most two 1-agents (its neighbors in the exploration graph). Therefore, after round 3, there are still at least $\left(\frac{4\alpha}{17} - \gamma\right) \cdot n$ more known 0-agents than known 1-agents. The algorithm uses these excess 0-agents for exploration in round 4. Since the exploration graph is now 3-regular, the adversary can use a greedy construction to select an independent set of at least one fourth of the explored nodes, containing at least $\left(\frac{\alpha}{17} - \frac{\gamma}{4}\right) \cdot n$ nodes. The number of remaining unknown 0-nodes at this point is $\left(\frac{\alpha}{17} - \beta - \gamma\right) \cdot n$, so the adversary can reveal $\min\left(\left(\frac{\alpha}{17} - \beta - \gamma\right) \cdot n, \left(\frac{\alpha}{17} - \frac{\gamma}{4}\right) \cdot n\right) = \left(\frac{\alpha}{17} - \beta - \gamma\right) \cdot n$ 0-agents. This gives regret $\left(\frac{\alpha}{17} - \beta - \gamma\right) \cdot n$ in round 4, bringing the total accumulated regret to $\frac{13\alpha}{17}n$. ■

The remaining lemmas handle the case $\alpha \geq \frac{1}{2}$. Recall that here, the per-round regret is equal to the number of $(1,1)$-teams that were selected.

Lemma 4. *The adversary has a strategy that ensures regret at least $\frac{k}{2}$ for each $\frac{1}{2} \leq \alpha \leq 1$.*

Proof The adversary reveals $\frac{k}{2}$ $(1,1)$-teams in round 1, giving regret $\frac{k}{2}$. ■

Note that by definition, $\frac{k}{2} = \frac{1-\alpha}{2} \cdot n$. Next, we show that the adversary has an alternate 2-round strategy which establishes a different linear lower bound on the regret.

Lemma 5. *The adversary has a strategy that ensures regret at least $\frac{3-4\alpha}{3} \cdot n$ for each $\frac{1}{2} \leq \alpha \leq 1$.*

Proof In round 1, the adversary reveals $\frac{\alpha}{3} \cdot n$ $(0,0)$-teams. The remaining teams are $\frac{\alpha}{3} \cdot n$ $(0,1)$-teams and $\frac{3-4\alpha}{6} \cdot n$ $(1,1)$-teams, giving regret $\frac{3-4\alpha}{6} \cdot n$. In round 2, the algorithm uses the $\frac{2\alpha}{3}$ known 0-agents to explore. It must explore members of at least $\frac{\alpha}{3}$ teams from round 1. The adversary can label one member each from $\frac{\alpha}{3}$ of these teams as a 0-agent, giving additional regret $\frac{3-4\alpha}{6} \cdot n$ in round 2, and making the total accumulated regret $\frac{3-4\alpha}{3} \cdot n$. ■

The bound from Lemma 5 is strictly better than the one from Lemma 4 for $\frac{1}{2} \leq \alpha < \frac{3}{5}$. Finally, we present a 3-round adversary strategy that establishes a third linear regret bound.

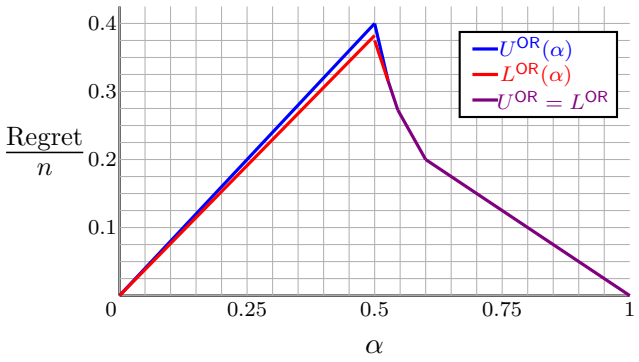


Figure 1: *Our regret bounds (Theorems 3 to 5) under the Strongest Link model, as functions of* $\alpha := \frac{n-k}{n}$*, the fraction of low-skill agents. The bounds match for* $\frac{10}{19} \leq \alpha \leq 1$.

Lemma 6. *The adversary has a strategy that ensures regret at least* $\frac{6-9\alpha}{4} \cdot n$ *for each* $\frac{1}{2} \leq \alpha \leq 1$.

Proof In round 1, the adversary reveals $\frac{\alpha}{4} \cdot n$ $(0,0)$-teams. The remaining teams are $\frac{\alpha}{2} \cdot n$ $(0,1)$-teams and $\frac{2-3\alpha}{4} \cdot n$ $(1,1)$-teams, giving regret $\frac{2-3\alpha}{4} \cdot n$. In round 2, the algorithm uses the $\frac{\alpha}{2} \cdot n$ known 0-agents to explore, exploring members from $\left(\frac{\alpha}{4} + \beta\right) \cdot n$ successful teams from round 1, for some $0 \leq \beta \leq \frac{\alpha}{4}$. The adversary can label one member from each of these teams as a 0-agent. Thus, the algorithm observes $\left(\frac{\alpha}{4} + \beta\right) \cdot n$ (0,0)-teams, $\left(\frac{\alpha}{2} - 2\beta\right) \cdot n$ (0,1)-teams, and $\left(\frac{2-3\alpha}{4} - \beta\right) \cdot n$ (1,1)-teams, giving the algorithm regret $\left(\frac{2-3\alpha}{4} - \beta\right) \cdot n$.

Note that every explored 0-agent in round 2 had been paired with a 1-agent in round 1. Therefore, the algorithm again has $\frac{\alpha}{2} \cdot n$ 0-agents with which to explore in round 3, and there are $\left(\frac{\alpha}{4} - \beta\right)$ unknown 0-agents. Note that the unresolved subgraph is bipartite after round 2, and at least half of the nodes explored by the algorithm must fall on the same side of this bipartition. Therefore, the adversary can reveal at least $\min\left(\frac{1}{2} \cdot \frac{\alpha}{2} \cdot n, \left(\frac{\alpha}{4} - \beta\right) \cdot n\right) = \left(\frac{\alpha}{4} - \beta\right) \cdot n$ 0-agents in round 3. This gives regret $\left(\frac{2-3\alpha}{4} + \beta\right) \cdot n$ in round 3, so the total regret is $\frac{6-9\alpha}{4} n$. ■

The bound from Lemma 6 is strictly better than those from Lemmas 4 and 5 for $\frac{1}{2} \leq \alpha < \frac{6}{11}$. Taking the pointwise maximum of the linear functions in the lemmas, we get Theorem 3.

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.

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[1].)
  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.

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.

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 7. *$\Delta_1 = d_1$, 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 7, 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 7 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 $U^{\mathsf{OR}}(\alpha) \cdot n$, where*

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

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 8. *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 repaired 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[2] 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 $(s_1^{00}, s_1^{01}, s_1^{11})$.

Lemma 9. *Conditioned on $s_1^{00}, s_1^{01}, s_1^{11}$, Algorithm 3 has regret at most*

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

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

Proof From Lemma 8 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 10. *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 7. 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 7, 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 $\alpha > \frac{1}{2}$. ■

To conclude, we get the piecewise-linear bound in Theorem 5 by maximizing the bound in Lemma 9 subject to the constraint in Lemma 10 (and using $s_t^{01} = \alpha n - 2s_t^{00}$, $s_t^{11} = s_t^{00} - (\alpha - \frac{1}{2})n$).

Recall that Lemma 9 establishes the following regret bound:

$$R(s^{00}, s^{01}, s^{11}) := \left\lfloor \frac{s^{01}+s^{11}}{s^{00}} \right\rfloor \cdot s^{11} + \min\left(s^{11}, (s^{01}+s^{11}) \mod s^{00}\right).$$

To obtain a more convenient form, we now rewrite R as a function of only s^{00}.

Lemma 11. *The regret bound can be expressed as a function of s^{00} as*

$$f_\alpha(s^{00}) := \left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot \left(\frac{n}{2} - \alpha n\right) + \min\left(\left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot s^{00}, \alpha n - s^{00}\right).$$

Proof Using that $s^{01} = \alpha n - 2s^{00}$ and $s^{11} = \frac{n}{2} - \alpha n + s^{00}$, we can simplify the regret expression as follows:

$$\begin{aligned}
R(s^{00}, s^{01}, s^{11}) &= \left\lfloor \frac{s^{01}+s^{11}}{s^{00}} \right\rfloor \cdot s^{11} + \min\left(s^{11}, (s^{01}+s^{11}) \mod s^{00}\right) \\
&= \left\lfloor \frac{n}{2s^{00}} - 1 \right\rfloor \cdot s^{11} + \min\left(s^{11}, \tfrac{n}{2} \mod s^{00}\right) \quad (\text{because } s^{01}+s^{11} = \tfrac{n}{2} - s^{00}) \\
&= \left\lfloor \frac{n}{2s^{00}} - 1 \right\rfloor \cdot s^{11} + \min\left(s^{11}, \tfrac{n}{2} - s^{00} \cdot \left\lfloor \frac{n}{2s^{00}} \right\rfloor\right) \\
&\qquad (\text{because } \tfrac{n}{2} \mod s^{00} = \tfrac{n}{2} - s^{00} \cdot \left\lfloor \tfrac{n}{2s^{00}} \right\rfloor) \\
&= \left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot s^{11} + \min\left(0, \tfrac{n}{2} - s^{11} - s^{00} \cdot \left\lfloor \frac{n}{2s^{00}} \right\rfloor\right) \\
&= \left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot \left(\tfrac{n}{2} - \alpha n + s^{00}\right) + \min\left(0, \alpha n - s^{00} - s^{00} \cdot \left\lfloor \frac{n}{2s^{00}} \right\rfloor\right) \\
&\qquad (\text{because } s^{11} = \tfrac{n}{2} - \alpha n + s^{00}) \\
&= \left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot \left(\tfrac{n}{2} - \alpha n\right) + \min\left(\left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot s^{00}, \alpha n - s^{00}\right) \\
&= f_\alpha(s^{00}).
\end{aligned}$$

■

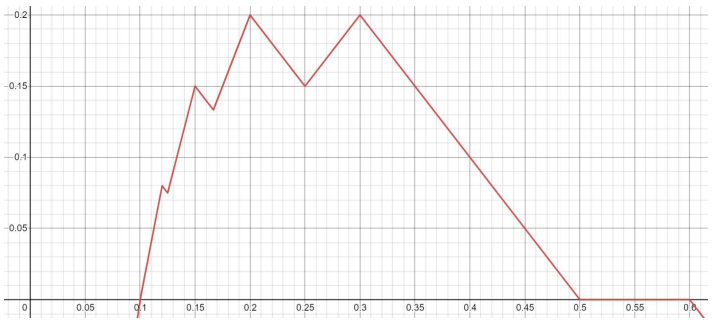


Figure 2: The function $f_\alpha(s^{00})$ for $\alpha = 0.6$. The local maxima occur at $0.12, 0.15, 0.2$, and 0.3 (i.e., at $\frac{\alpha}{z}$ for $z \in \{2, 3, 4, 5\}$).

Lemma 12. *For any $\alpha \geq \frac{1}{2}$, $R(s^{00}, s^{01}, s^{11})$ is maximized when $\frac{s^{00}}{n} = \frac{\alpha}{z}$ for some integer $z \geq 2$. This implies that*

$$R(s^{00}, s^{01}, s^{11}) \leq \max_{z \in \mathbb{Z}_{>0}} \left\{(z-1) \cdot \left(\tfrac{1}{2} - \alpha + \tfrac{\alpha}{z}\right)\right\} \cdot n = \max_{z \in \mathbb{Z}_{>0}} \left\{\tfrac{z(z-1) - 2(z-1)^2\alpha}{2z}\right\} \cdot n. \quad (1)$$

Proof The function $f_\alpha(s^{00})$ derived in Lemma 11 has a saw-tooth shape (see Fig. 2).

We first prove that at any global maximum of $f_\alpha(\cdot)$, the two terms under the minimum must be equal, i.e., $\left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot s^{00} = \alpha n - s^{00}$. We do so by distinguishing two cases:

1. If $\frac{n}{2s^{00}}$ is not an integer, then $\left\lfloor \frac{n}{x} \right\rfloor$ is constant for x in a small interval around s^{00}. In this interval, the first term of the minimum is increasing in s^{00} and the second decreasing in s^{00} (while the term before the minimum is constant in this interval). Therefore, if the two terms under the minimum were not equal, by increasing or decreasing s^{00} slightly, the minimum could be increased.

2. If $\frac{n}{2s^{00}}$ is an integer, then we first argue that the minimum must equal the first term. The reason is as follows: when s^{00} is decreased slightly, $\left\lfloor \frac{n}{2s^{00}} \right\rfloor$ stays constant, and the s^{00} factor decreases slightly (while the term outside the minimum stays constant). On the other hand, $\alpha n - s^{00}$ increases slightly; if the first term were strictly larger than the minimum, then the minimum could be increased by slightly decreasing s^{00}.

   On the other hand, when $\frac{n}{2s^{00}}$ is an integer, the first term under the minimum equals $\frac{n}{2}$, while the second equals $\alpha n - s^{00}$. Because αn is the number of 0-agents, $\alpha n - s^{00}$ is the number of pairs including at least one 0-agent, which is at most the total number of pairs, $\frac{n}{2}$. Therefore, the minimum must also equal the second term.

Thus, we have shown that $\left\lfloor \frac{n}{2s^{00}} \right\rfloor \cdot s^{00} = \alpha n - s^{00}$ at any global maximum of f_α. Rearranging this equation, we obtain that $\alpha = \left\lfloor \frac{n}{2s^{00}} + 1 \right\rfloor \frac{s^{00}}{n}$ at any local maximum. Note that $s^{00} \leq \frac{\alpha n}{2}$, so $\left\lfloor \frac{n}{2s^{00}} + 1 \right\rfloor$ is an integer, and at least 2. Thus, any global maximum occurs when $\frac{s^{00}}{n} = \frac{\alpha}{z}$ for some integer $z \geq 2$. To derive the forms in Inequality (1), we equate the two terms in the minimum expression for $s^{00} = \frac{\alpha n}{z}$. This gives us that $\left\lfloor \frac{z}{2\alpha} \right\rfloor \cdot \frac{\alpha n}{z} = \alpha n \cdot \frac{z-1}{z}$, so $\left\lfloor \frac{z}{2\alpha} \right\rfloor = z - 1$. Plugging these expressions into f_α, we find that

$$f_\alpha(\tfrac{\alpha n}{z}) \;=\; \left\lfloor \tfrac{z}{2\alpha} \right\rfloor \cdot \left(\tfrac{n}{2} - \alpha n\right) + \left\lfloor \tfrac{z}{2\alpha} \right\rfloor \cdot \tfrac{\alpha n}{z} \;=\; \left\lfloor \tfrac{z}{2\alpha} \right\rfloor \cdot \left(\tfrac{1}{2} - \alpha + \tfrac{\alpha}{z}\right) \cdot n \;=\; (z-1) \cdot \left(\tfrac{1}{2} - \alpha + \tfrac{\alpha}{z}\right) \cdot n.$$

This completes the proof of the lemma. ■

Using Lemma 10, we have the additional constraint that $z \leq 5$. In this case, Eq. (1) simplifies to

$$\max\left\{0, \tfrac{1-\alpha}{2}, \tfrac{3-4\alpha}{3}, \tfrac{6-9\alpha}{4}, \tfrac{10-16\alpha}{5}\right\} \cdot n.$$

By determining which of these linear functions is largest for each $\frac{1}{2} < \alpha \leq 1$, we obtain the bound from Theorem 5.

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* $L^{\mathsf{AND}}(n,k) := n-k$.

We will prove this theorem by describing a greedy policy of the adversary that establishes the lower regret bound $n-k$ for the weakest link model. Note that the undiscovered subgraph for this model consists entirely of unsuccessful edges. Thus, the 1-agents form an independent set in this subgraph.

A *labeling* is a function $\lambda : [n] \to \{0,1\}$ assigning the skill of each agent. It is *viable* if $\sum_{i=1}^{n} \lambda(i) = k$ (it assigns exactly k 1-agents) and it agrees with all previously revealed edges.

Suppose that G is an undiscovered subgraph on which the algorithm is about to play a matching M. Further suppose that there are j undiscovered 1-agents in G. Then, a subset $S \subseteq M$ is a *revealable set* if there exists a viable labeling under which the edges in S are successful, and the edges in $M \setminus S$ are unsuccessful.

A *minimal revealable set* is a revealable set for which no proper subset is a revealable set. Note that whenever $|M| \geq \frac{j}{2}$, any $\frac{j}{2}$-subset of M is revealable: all remaining nodes can be 0-agents according to the previously revealed information. Therefore, a *minimal revealable set* exists. Note that if $G \cup M$ has an independent set of size j, then $\emptyset$ is the unique minimal revealable set. With this terminology, we can describe the policy of the adversary in two consecutive steps per round:

1. Iterate over the *explored agents* u, i.e., the yet-undiscovered agents that are paired with a discovered 1-agent in this round. If revealing u as a 0-agent still admits a viable labeling of the remaining agents, the adversary reveals u as a 0-agent. Otherwise, u is revealed as a 1-agent.

2. Let G denote the resulting undiscovered graph after labeling all nodes whose type could be deduced from step 1. The adversary chooses a minimal revealable set S for G, labels these edges as successful, and the remaining edges as unsuccessful.

To argue the regret bound, we make use of the following three lemmas.

Lemma 13. *Let u be a 1-agent explored by being paired with a known 1-agent. Let w be a 0-agent that was discovered through its connection to u. Then, in all viable labelings, w is connected to at least two 1-agents.*

Proof For the sake of contradiction, suppose that there is a viable labeling λ in which w's only 1-agent connection is to u. Consider the alternative labeling λ' which switches the labels of u and w in λ, but keeps all other labels the same. This labeling is also viable at this point since neither w (by assumption) nor u (it is undiscovered) has been paired with any other 1-agent. However, the existence of λ' contradicts the adversary's necessity to reveal u as a 1-agent. ■

Lemma 14. *Let G be an undiscovered subgraph from step 2 containing j 1-agents. Let S be a minimal revealable set, and let w be a 0-agent that was discovered through its connection to an endpoint u of an edge in S. Then, in all viable labelings (assigning type 1 to all*[3] *endpoints*

[3]Note that no edge in S can have been played in a previous round — otherwise, revealing it as a successful edge would not leave any viable labelings, as it would contradict the earlier revelation as unsuccessful. This

of edges in S), w is connected to at least two 1-agents.

Proof For the sake of contradiction, suppose that there is a labeling λ for which the only 1-agent that w is connected to is u. Let v denote the other endpoint of u's revealed 1-edge (so $(u, v) \in S$). Finally, let I denote a set of $j - 2|S|$ 1-agents (given by λ) in the resulting undiscovered subgraph (guaranteed to exist because S is revealable). By assumption, w is connected to neither v nor any agent in I. Additionally, no agent in I is connected to v (because the agents in I are undiscovered after (u, v) is revealed as a 1-edge). Therefore, $I \cup \{w, v\}$ is an independent set of size $j - 2|S| + 2$ in G. However, this means that $S \setminus \{(u, v)\}$ is a revealable set (a viable labeling λ' is obtained by starting with λ, and switching the label of u to 0 while switching the label of w to 1), contradicting the minimality of S. ■

Lemma 15. *Suppose that the adversary (using this policy) has just revealed the first 1-edge(s). Then, in any viable labeling λ, each undiscovered 0-agent is connected to at least one undiscovered 1-agent.*

Proof Let S be the minimal revealable set that was just revealed by the adversary. Fix any viable labeling λ, and let I denote the set of undiscovered 1-agents in λ. For the sake of contradiction, suppose that there is an undiscovered 0-agent w that is not paired with an agent in I. Then, $I \cup \{w\}$ is an independent set of size $j - 2|S| + 1$. If $(u, v) \in S$ is any successful edge that was just revealed, then u has no edges to I (by viability of λ) nor to w (or w would have been discovered as a 0-agent). Therefore, $I \cup \{w, u\}$ is an independent set of size $j - 2|S| + 2$, so $S \setminus \{(u, v)\}$ is a revealable set; the viable labeling λ' witnessing this is obtained from λ by switching the label of w to 1 and the label of v to 0. This contradicts the minimality of S. ■

Using these lemmas, we can show that any policy incurs a regret of at least $n - k$ under this adversarial policy.

Proof of Theorem 6 It suffices to argue that, at the time of its discovery, each 0-agent must have been paired with at least two 1-agents in any viable labeling. (Lemma 15 guarantees that all 0-agents are connected to a 1-agent after the first discovery, so once all 1-agents are discovered, all 0-agents will also have been discovered.) Lemmas 13 and 14 ensure this for all 0-agents that are discovered by having a connection to a 1-agent that is discovered. Lemma 15 ensures this for all 0-agents u that are explored by a known 1-agent w. This is because u will be connected to w and the guaranteed undiscovered 1-agent v, which it was connected to earlier (implying that $v \neq w$).

Therefore, the algorithm plays at least $2(n - k)$ $(0, 1)$-edges, leading to regret at least $n - k$. ■

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

would contradict the definition of S being revealable.

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 $T(n, \frac{n}{k})$ [22, 1] has regret $\frac{1}{2}k(n-k)$. Similarly, the performance of Exponential Cliques in the worst case is also much worse.

Lemma 16. 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 to it 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 Appendix A, and we visualize the first 5 matchings in the construction for $n = 10$ and $n = 12$ in Fig. 3.

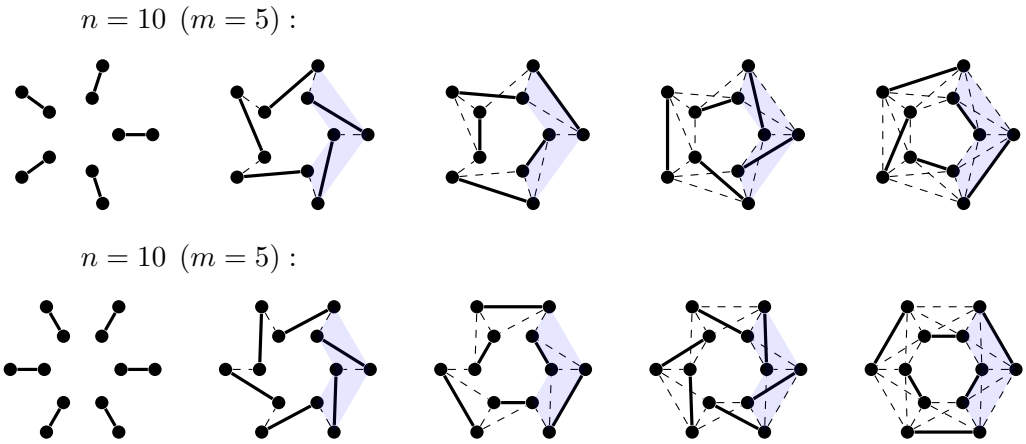


Figure 3: *The first five rounds (i.e., Phases 0 and 1) of* Ring Factorization *on 10 (top) and 12 (bottom) agents. The last four matchings illustrate the general matching sequence for cycles in intermediate phases; the blue highlighted section of each matching is repeated based upon the size of the cycle.*

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

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.)
    Play known $(0,0)$- and $(1,1)$-teams, and continue playing matchings according to Ring Factorization on the remaining n' agents.

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 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 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, 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] Martin Aigner. Turán's graph theorem. *The American Mathematical Monthly*, 102(9):808–816, 1995.

[2] Moshe Babaioff, Michal Feldman, and Noam Nisan. Combinatorial agency. In *Proc. 7th ACM Conf. on Electronic Commerce*, page 18–28, 2006.

[3] Guillaume Carlier and Ivar Ekeland. Matching for teams. *Economic Theory*, 42:397–418, 02 2010.

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

[5] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *ICML*, 2013.

[6] Richard Combes, M. Sadegh Talebi, Alexandre Proutiere, and Marc Lelarge. Combinatorial bandits revisited. In *Proc. 29th Advances in Neural Information Processing Systems*, page 2116–2124, 2015.

[7] Nikhil R Devanur and Kamal Jain. Online matching with concave returns. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 137–144, 2012.

[8] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, 2012.

[9] Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex online problems. In *International conference on machine learning*, pages 100–108. PMLR, 2014.

[10] Yanjun Han, Yining Wang, and Xi Chen. Adversarial combinatorial bandits with general non-linear reward functions. In *International Conference on Machine Learning*, pages 4030–4039, 2021.

[11] Chamsi Hssaine and Siddhartha Banerjee. Information signal design for incentivizing team formation. In *Proc. 14th Conference on Web and Internet Economics (WINE)*, 2018.

[12] Ramesh Johari, Vijay Kamble, Anilesh K. Krishnaswamy, and Hannah Li. Exploration vs. exploitation in team formation. In *Proc. 14th Conference on Web and Internet Economics (WINE)*, 2018.

[13] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, Claire Vernade, and Zheng Wen. Stochastic rank-1 bandits. In *Artificial Intelligence and Statistics*, pages 392–401. PMLR, 2017.

[14] Johannes Kirschner, Tor Lattimore, and Andreas Krause. Information directed sampling for linear partial monitoring. In *Conference on Learning Theory*, pages 2328–2369. PMLR, 2020.

[15] Jon Kleinberg and Maithra Raghu. Team performance with test scores. In *Proc. 16th ACM Conf. 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] Nadav Merlis and Shie Mannor. Tight lower bounds for combinatorial multi-armed bandits. In *Conference on Learning Theory*, pages 2830–2857. PMLR, 2020.

[18] Arun Rajkumar, Koyel Mukherjee, and Theja Tulabandhula. Learning to partition using score based compatibilities. In *Proc. 16th Intl. Conf. on Autonomous Agents and Multiagent Systems*, page 574–582, 2017.

[19] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Advances in Neural Information Processing Systems*, 27, 2014.

[20] Flore Sentenac, Jialin Yi, Clement Calauzenes, Vianney Perchet, and Milan Vojnovic. Pure exploration and regret minimization in matching bandits. In *ICML21*, pages 9434–9442, 2021.

[21] Adish Singla, Eric Horvitz, Pushmeet Kohli, and Andreas Krause. Learning to hire teams. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[22] Paul Turán. Egy gráfelméleti szélsoértékfeladatról (on an extremal problem in graph theory). *Mat. Fiz. Lapok*, 48(3):436–453, 1941.

[23] Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pages 5114–5122. PMLR, 2018.

[24] Julian Zimmert and Yevgeny Seldin. Factored bandits. *Advances in Neural Information Processing Systems*, 31, 2018.

A Ring Factorization: Detailed Construction

Let $m := \frac{n}{2}$, and arbitrarily label the n agents as $u_0, u_1, \ldots, u_{m-1}$ and $v_0, v_1, \ldots, v_{m-1}$. In Fig. 3, the u_i are the nodes on the inner ring; the pair of vertices with matching indices form a "column" of the ring. We separate the factorization into $\left\lceil \frac{m+1}{2} \right\rceil$ phases (numbered starting from 0). In each phase i, we select matchings to exactly add the edges connecting agents whose subscripts are i apart (modulo m). Note that agent subscripts differ by at most $\lfloor \frac{m}{2} \rfloor$, so this accounts for all edges.

Phase 0: This phase lasts for 1 round

Select the edges $\{(u_0, v_0), (u_1, v_1), \ldots, (u_{m-1}, v_{m-1})\}$.

Phase i ($1 \leq i < \frac{m}{2}$): Each of these phases lasts 4 rounds.

Consider the graph on $[m]$ with an edge between two indices iff they are exactly i apart modulo m. This graph is a union of $\gcd(m, i)$ disjoint cycles, each containing $j := \frac{m}{\gcd(m,i)} \geq 3$ indices.

It suffices to define a procedure for selecting perfect matchings within each of these cycles. Focus on one cycle of length j, and relabel the corresponding $2j$ agents as $u'_0, u'_1, \ldots, u'_{j-1}$ and $v'_0, v'_1, \ldots, v'_{j-1}$. There are two cases based on whether j is even or odd.

If j is even, the factorization uses the following four matchings:

Round 1: $\left\{(u'_0, v'_1), (u'_1, v'_2), \ldots, (u'_{j-1}, v'_0)\right\}$

Round 2: $\left\{(u'_0, u'_1), (u'_2, u'_3), \ldots, (u'_{j-2}, u'_{j-1})\right\} \cup \left\{(v'_1, v'_2), (v'_3, v'_4), \ldots, (v'_{j-1}, v'_0)\right\}$

Round 3: $\left\{(v'_0, u'_1), (v'_1, u'_2), \ldots, (v'_{j-1}, u'_0)\right\}$

Round 4: $\left\{(u'_1, u'_2), (u'_3, u'_4), \ldots, (u'_{j-1}, u'_0)\right\} \cup \left\{(v'_0, v'_1), (v'_2, v'_3), \ldots, (v'_{j-2}, v'_{j-1})\right\}$

If j is odd, the factorization uses the following four matchings:

Round 1: $\left\{(u'_0, v'_1), (u'_1, v'_2), \ldots, (u'_{j-1}, v'_0)\right\}$

Round 2: $\left\{(v'_0, u'_1)\right\} \cup \left\{(u'_2, u'_3), (u'_4, u'_5), \ldots, (u'_{j-1}, u'_0)\right\} \cup \left\{(v'_1, v'_2), (v'_3, v'_4), \ldots, (v'_{j-2}, v'_{j-1})\right\}$

Round 3: $\left\{(v'_0, v'_1), (u'_1, u'_2)\right\} \cup \left\{(v'_2, u'_3), (v'_3, u'_4), \ldots, (v'_{j-1}, u'_0)\right\}$

Round 4: $\left\{(v'_1, u'_2), (u'_0, u'_1)\right\} \cup \left\{(u'_3, u'_4), \ldots, (u'_{j-2}, u'_{j-1})\right\} \cup \left\{(v'_2, v'_3), (v'_4, v'_5), \ldots, (v'_{j-1}, v'_0)\right\}$

Since each node is part of exactly one cycle, taking the union over all cycles gives a perfect matching in each of the four rounds.

Phase $\frac{m}{2}$ (if m is even): This phase lasts 2 rounds.

Round 1: $\left\{(u_0, u_{\frac{m}{2}}), (u_1, u_{\frac{m}{2}+1}), \ldots, (u_{\frac{m}{2}-1}, u_{m-1})\right\} \cup \left\{(v_0, v_{\frac{m}{2}}), (v_1, v_{\frac{m}{2}+1}), \ldots, (v_{\frac{m}{2}-1}, v_{m-1})\right\}$

Round 2: $\left\{(u_0, v_{\frac{m}{2}}), (u_1, v_{\frac{m}{2}+1}), \ldots, (u_{\frac{m}{2}-1}, v_{m-1})\right\} \cup \left\{(v_0, u_{\frac{m}{2}}), (v_1, u_{\frac{m}{2}+1}), \ldots, (v_{\frac{m}{2}-1}, u_{m-1})\right\}$

B Details of the Ring Factorization with Repairs Algorithm

Here, we present a proof of Theorem 7, along with the missing details of the algorithm Ring Factorization with Repairs.

We first note that it suffices to prove a regret bound of $n - k + \lfloor \frac{k}{4} \rfloor$. For suppose that $k > \frac{n}{2}$. In round 1, there can be at most $n - k$ unsuccessful teams, meaning that some $(1,1)$-teams must be revealed. An optimal matching then consists of these initially revealed $(1,1)$-teams, plus an optimal matching on the unresolved agents. Since the unresolved agents were each part of an unsuccessful team in round 1, at least half of them are 0-agents. Notably, the number of 1-agents that remain is at most $n - k$. Of course, the initial $(1,1)$-teams did not contribute any regret, so the instance has been reduced to a smaller (n', k')-instance, with $n' - k' = n - k$ and $k' \leq n - k$. If we can obtain a regret bound of $n' - k' + \left\lfloor \frac{k'}{4} \right\rfloor$, then for the original instance, it results in a regret bound of

$$n' - k' + \left\lfloor \frac{k'}{4} \right\rfloor \leq n - k + \left\lfloor \frac{n-k}{4} \right\rfloor .$$

Recall that all the regret incurred by an algorithm can be charged to the $(0,1)$-pairs that it plays. More precisely, any two $(0,1)$-pairs played in a round could be re-paired into a $(0,0)$-pair and a $(1,1)$-pair, which would contain one successful pair instead of zero. Therefore, we can charge half a unit of regret to each $(0,1)$-pair played by an algorithm. Hence, the total regret is bounded by

$$\text{Regret} \leq \frac{1}{2} \cdot \#\left\{\text{explored } (0,1)\text{-pairs}\right\} = \frac{1}{2} \sum_{\text{1-agent } v} \#\left\{(0,1)\text{-teams including } v\right\}. \tag{2}$$

We will use this viewpoint, and analyze the regret for each 1-agent separately, by analyzing how many 0-agents it was paired with. As described in the proof sketch, ideally, we would like to prove that each 1-agent is paired only with two 0-agents. This is not quite true, and we need a more intricate argument to obtain the (slightly weaker) bound of the theorem.

The sketch of the Ring Factorization with Repairs Algorithm exhibits a natural recursive structure, making it amenable to an inductive argument. In particular, we argue by induction on n that Ring Factorization with Repairs on n agents (n even) with $k \leq n$ 1-agents (k even) locates an optimal matching after incurring regret at most $n - k + \lfloor \frac{k}{4} \rfloor$. Along the way, we flesh out the details of the repairing step.

The base case $n \in \{0, 2\}$ is trivial, as there is a unique matching that incurs regret zero. The inductive hypothesis states that for some $n \geq 4$, for every even $n' < n$ and every even k' with $0 \leq k' \leq n'$, the Ring Factorization with Repairs Algorithm on n' agents, k' of whom are 1-agents, determines an optimal matching after incurring regret at most $n' - k' + \left\lfloor \frac{k'}{4} \right\rfloor$.

For the inductive step, consider an instance with n agents, k of whom are 1-agents. If $k \leq 1$, then any matching procedure incurs no regret. In particular, the Ring Factorization with Repairs algorithm satisfies the upper regret bound. Therefore, it suffices to reason

about instances with $k \geq 2$. Since the Ring Factorization with Repairs algorithm initially selects matchings according to the ring factorization (a 1-factorization), it must in some round play an edge between two 1-agents for the first time. We perform a case analysis based on the time of this first 1-edge discovery below.

The inductive step relies on a slightly nuanced way to calculate the total regret of the procedure. Suppose that after round r, the algorithm is able to remove z discovered 0-agents (by pairing them up permanently) and w 1-agents (also pairing them up permanently), with the following additional properties:

- By adding edges which can be deduced to be unsuccessful from the observed outcomes, the induced subgraph on the remaining agents can be made isomorphic to an intermediary stage of a smaller ring factorization.
- There is at most one discovered 1-agent that is not removed.

The 1-agents that the algorithm removes will never again be paired with a 0-agent, so they have already accumulated their share of the regret. The remaining $k - w$ 1-agents have cumulatively been incident on $r(k - w) - x$ (0,1)-edges, where $x \in \{0, 1\}$ is the number of removed 1-agents that the discovered, un-removed 1-agent had been paired with (if there was such an agent). Ring Factorization with Repairs will ensure that in the sub-factorization, the $k - w$ 1-agents will each be incident on exactly r $(0, 1)$-edges. Most of the technical work of the proof goes into proving the following lemma, which we will do below.

Lemma 17. *Each removal of the form described above satisfies*

$$\frac{1}{2} \cdot \sum_{\substack{\text{removed} \\ \text{1-agent } v}} \#\{(0,1)\text{-teams including } v\} \leq z + \left\lfloor \frac{w}{4} \right\rfloor + \frac{x}{2}. \tag{3}$$

For the remaining regret, we apply the induction hypothesis to the strictly smaller ring factorization on the remaining agents. The induction hypothesis guarantees a regret of

$$\begin{aligned} \text{Regret of Sub-Factorization} &\leq (n - (z + w)) - (k - w) + \left\lfloor \frac{k - w}{4} \right\rfloor \\ &\leq n - k + \left\lfloor \frac{k}{4} \right\rfloor - z - \left\lfloor \frac{w}{4} \right\rfloor. \end{aligned}$$

Using the bounds from Lemma 17 and the induction hypothesis, we can now decompose and bound the regret from Equation (2) as follows:

$$\begin{aligned} \text{Regret} &= \tfrac{1}{2} \sum_{\substack{\text{removed} \\ \text{1-agent } v}} \#\{(0,1)\text{-teams including } v\} + \text{Regret of Sub-Factorization} - \frac{x}{2} \\ &\leq \left(z + \left\lfloor \frac{w}{4} \right\rfloor + \frac{x}{2}\right) + \left(n - k + \left\lfloor \frac{k}{4} \right\rfloor - z - \left\lfloor \frac{w}{4} \right\rfloor\right) - \frac{x}{2} \\ &= n - k + \left\lfloor \frac{k}{4} \right\rfloor. \end{aligned}$$

This completes the proof of the induction step, and thus the theorem.

In the remainder of this section, we prove Lemma 17.

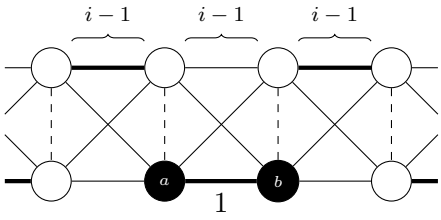
Proof of Lemma 17 We perform a detailed case analysis based on the time of the first discovery of a successful edge. We always denote the successful edge by (a, b).

1. Discovery during phase 0:

   Removing the endpoints of the discovered edge gives an intermediary stage of the sub-factorization on $n-2$ agents with $(k-2)$ 0-agents. In this case, we have $z = 0, w = 2$, and $x = 0$. The removed 1-agents were incident to no $(0, 1)$-edges, so Equation (3) is satisfied.

2. Discovery in round 4 of phase i $(1 \leq i < \frac{m}{2})$:

   We use the following diagram to represent the local area of the ring around the discovered successful edge.

   Here (and in all subsequent such diagrams), the edges represent pairs explored during phase i. The bolded matching was the last one played, i.e., the one that resulted in the 1-edge discovery. The top brackets indicate the presence of $i-1$ columns of agents between these matched columns. All of the missing agents in the labeled gaps are deduced to be 0-agents by their proximity to the discovered 1-edge. Discovered 1-agents are black, deduced 0-agents are white, and unknown agents are gray (in the later diagrams).

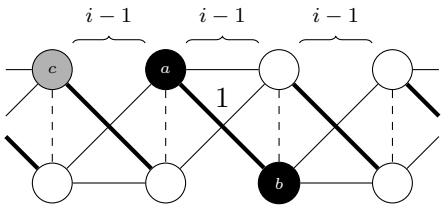
   The algorithm removes the neighborhood of b; i.e., the i columns to the left of b (including a) and the i columns to the right of b. Notice that it does not remove the columns strictly to the left of a, even though they can be deduced to consist of 0-agents. Instead, the algorithm can deduce that all the edges between these agents and the agents to the right of b are unsuccessful, and can use this deduction to patch the gap in the ring. This allows the algorithm to obtain an intermediary stage of a sub-factorization.

   Both a and b had been paired with $4i$ 0-agents. The number of removed 0-agents is $z = 4i$, and two 1-agents (a and b) are removed, so $w = 2$. $x = 0$ because no known 1-agents is retained. Substituting these values, we see that Equation (3) is satisfied. Note that the same argument also handles the case where (a, b) is a diagonal edge in our diagram, as can happen in round 4 in an odd cycle.

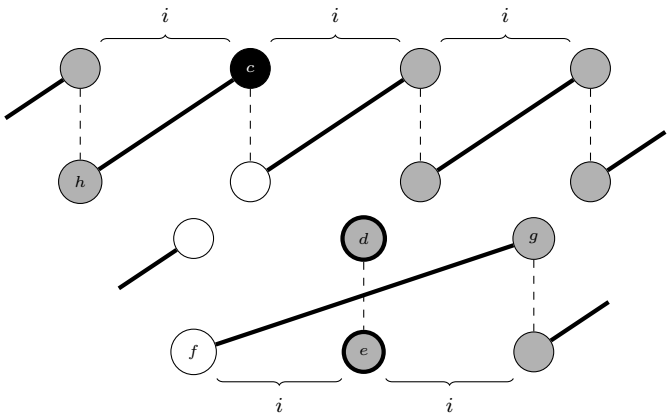
3. Discovery in round 3 of phase i $(1 \leq i < \frac{m}{2})$:

   There are 3 sub-cases to consider.

(a) In the first sub-case, one endpoint (w.l.o.g. b) of the edge (a, b) had already been connected to both vertices on the other side, while the other (a) had not. Let c be the node on the other side that a had not been connected to, and assume further that c was paired with the 0-agent in the same column as a in the previous round. Note that this subcase handles discovery in any even cycle, and all but two possible edge discoveries in an odd cycle. One such arrangement is shown below.



The algorithm plays the final matching as prescribed by Ring Factorization. The edge between a and c will identify the type of c. If c is a 0-agent, we can appeal to Case 2. Thus, we are left to handle the case in which c is a 1-agent. In that case, let d and e be the agents in the column $i+1$ columns to the right of b in the original ring. The algorithm removes the neighborhood of b. In the remaining subgraph, the algorithm can deduce all unsuccessful edges needed to obtain a sub-factorization, except for the edges (c, d) and (c, e). Thus, the algorithm needs to do more work in the next round in order to "catch up" to the sub-factorization. In the next round, the algorithm uses a and b to explore d and e (pairing up their prescribed neighbors), while pairing up the remaining agents according to the sub-factorization as shown below. In this figure, notice that the top and bottom parts are "interleaved" — for example, the nodes d and e are i columns to the right of c, and are the immediate left neighbors of the gray nodes shown above them and slightly to the right.

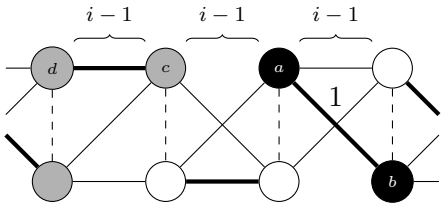


If both d and e are 0-agents, then the algorithm can deduce that (c, d), (c, e), (f, d), and (e, g) are unsuccessful edges, and has caught up to an intermediary stage of the ring factorization (forgetting about the explored (e, f) edge). In total, a was paired with $4i$ 0-agents (all $i + 2$ neighbors except for b and c) and b was paired with $4i + 1$ 0-agents (all $4i + 2$ neighbors except a), while $z = 4i$, $w = 2$, and $x = 1$, so Inequality (3) is satisfied.

If one of d, e is a 1-agent (w.l.o.g. d; they cannot both be 1-agents), then all of the i columns to the right of d, e consist of 0-agents. There are two further cases to consider.

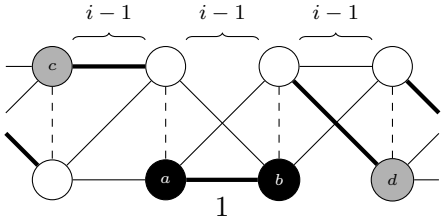
- If h is a 0-agent, then the algorithm can further remove the $2i + 1$ columns surrounding c. Each missing edge in the sub-factorization has one endpoint that is known to be a 0-agent, so we can appeal to the inductive hypothesis. Overall, c and d were each paired with $4i + 1$ 0-agents, and a and b were paired with a total of $8i$ 0-agents. In addition, $z = 8i$, $w = 4$, and $x = 0$, so Inequality (3) is satisfied.
- If h is a 1-agent, then the algorithm can remove $i - 1$ columns to the left of h, as well as the $i + 2$ columns between (and including) the columns of h and c. Note that the missing edges in the sub-factorization each have an endpoint that is a deduced 0-agent, so they can be added to obtain an intermediary round of a smaller ring factorization. In all, c was paired with $4i$ 0-agents, h was paired with $(4i + 1)$ 0-agents, and a and b were paired with a total of $8i$ 0-agents. In addition, $z = 8i$, $w = 4$, and $x = 0$, so Inequality (3) is satisfied.

(b) In the second sub-case, as in case (a), b had already been connected to both vertices on the other side, while a had not. Different from case (a), we assume that c was paired with an agent to its *left* in the previous round. This case cannot occur in an even cycle, and can occur in exactly one place in an odd cycle. The diagram for this case is shown below.



If the edge (c, d) is unsuccessful, we can appeal to Case 3(a). Otherwise, the edge (c, d) is successful, so d is identified as a 1-agent. The algorithm then removes the neighborhoods of c and b. The i columns of 0-agents to the left of d allow the algorithm to deduce all necessary 0-edges to obtain an intermediary state of the sub-factorization. Each of the $w = 4$ removed 1-agents was paired with $4i - 1$ 0-agents, $z = 8i - 2$, and $x = 0$, so Inequality (3) is satisfied.

(c) In the third sub-case, neither a nor b had connected to both vertices on the other side. This case cannot occur in an even cycle, and can occur in exactly one place in an odd cycle. The diagram for this case is shown below.

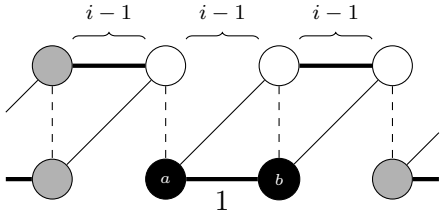


In the last round, the algorithm plays the matching prescribed by RING FACTORIZATION. If both c and d are 0-agents, we can appeal to case 2 of the proof. If exactly one of c, d is a 1-agent, we can appeal to Case 3(a). If both c and d are 1-agents, the algorithm removes the neighborhoods of a and d, and patches the ring as in Case 2. a and b were each paired with $4i-1$ 1-agents, and c and d were paired with $4i$ 1-agents. In addition, $z = 8i-2$, $w = 4$, and $x = 0$, so Inequality (3) is satisfied.

4. Discovery in round 2 of phase i ($1 \leq i < \frac{m}{2}$):

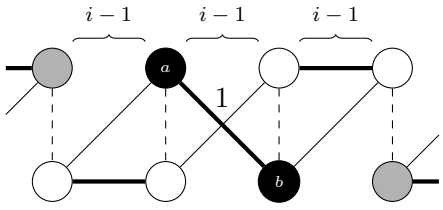
   There are two sub-cases to consider.

   (a) In the first sub-case, exactly one endpoint of the discovered successful edge (w.l.o.g. b) was connected to a vertex on the other side. This subcase handles discovery in any even cycle, and all but one possible edge discovery in an odd cycle. One such arrangement is shown below.



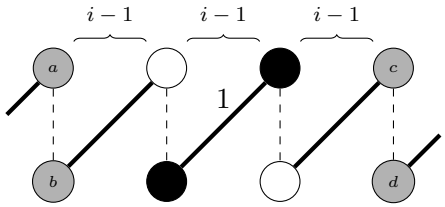
   The algorithm removes b's column, along with the i columns to its left and the $i-1$ columns to its right. Note that the $i-1$ columns of known 0-agents to the left of a and the known 0-agent with whom b was paired in the first round of this phase allow the algorithm to deduce all necessary 0-edges to obtain an intermediary stage of the sub-factorization. a and b were each paired with $4i-2$ 0-agents, $z = 4i-2$, $w = 2$, and $x = 0$, so Inequality (3) is satisfied.

   (b) In this sub-case, both a and b had been connected to an agent on their other side. This case cannot occur in an even cycle, and can occur in exactly one place in an odd cycle. The diagram for this case is shown below.

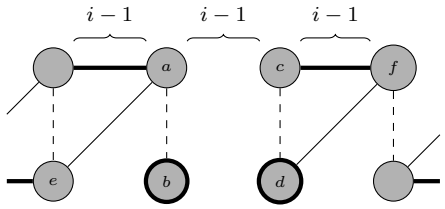


   Again, the algorithm removes b's column, along with the i columns to its left and the $i-1$ columns to its right. The $i-1$ columns of 0-agents to the left of a, along with the known 0-agents with whom a and b were paired in the first round of this phase, allow the algorithm to deduce all necessary 0-edges to obtain an intermediary stage of the sub-factorization. a and b were each paired with $4i-2$ 0-agents, $z = 4i-2$, $w = 2$, and $x = 0$, so Inequality (3) is satisfied.

5. Discovery in round 1 of phase i ($1 \leq i < \frac{m}{2}$): Our diagram is as follows; note that in this case, we do not use a and b to denote the discovered successful edge:



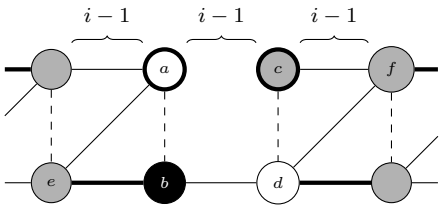
The algorithm begins by removing the column of the right 1-agent, as well as the i columns to its left and the $i-1$ columns to its right. At this point, the algorithm is almost left with an intermediary stage of a sub-factorization; however, it cannot deduce whether the edge (b, c) is successful (which is necessary to recover a sub-factorization). To "catch up" to the sub-factorization, the algorithm does some exploration with the removed 1-agents. In the second round, Ring Factorization prescribes a pairing of (a, c) or (b, d).[4] The algorithm uses the removed 1-agents to explore both endpoints of this edge, while playing the rest of the matching according to the sub-factorization. One possibility is shown below; by the footnote, it suffices w.l.o.g. to focus on this case.



If b is a 0-agent, then the algorithm can deduce the missing edge (b, c), and is back at an intermediary stage of the sub-factorization. The removed 1-agents were each matched to at most $4i-2$ 0-agents, $z = 4i-2$, $w = 2$, and $x \geq 0$, so Inequality (3) is satisfied.

Thus, it remains to handle cases in which b is a 1-agent. If the edge (c, f) is successful, then the algorithm removes b's column, along with the i columns to its right and $i-2$ columns to its left. The originally removed 1-agents were paired with a total of $8i-5$ 0-agents, b was paired with $4i-2$ 0-agents, and c with $4i-3$ 0-agents. In addition, $z = 8i-6$, $w = 4$, and $x = 1$, so Inequality (3) is satisfied. A similar procedure handles the cases in which d is a 1-agent, or in which e's edge in round 2 is successful. If none of these cases arises, the algorithm proceeds with the next round, as shown in the following diagram.

[4] In an even cycle, all round-2 edges are within the same ring. In an odd cycle, there will be one "diagonal" edge connecting nodes from different rings played in round 2. However, since the exploration graph is rotationally symmetric after round 1, the algorithm is free to choose the location of this diagonal edge; in particular, it places the edge between columns other than b's and c's.



If c is revealed to be a 0-agent, then the algorithm can deduce the missing edge (b, c) to be unsuccessful, and is back at an intermediary stage of the sub-factorization. In this case, the two originally removed 1-agents had been matched to a total of $8i-3$ 0-agents, $z = 4i - 2$, $w = 2$, and $x = 1$, so Inequality (3) is satisfied. We are left to handle cases where c is a 1-agent, implying that the $i-1$ columns to its right only contain 0-agents. Note that because we are in the case where the edge (c, f) was unsuccessful, this implies that f was a 0-agent.

If e is a 0-agent (which is revealed by the edge with b), the algorithm removes b's column, along with the $i - 1$ columns to its left and the i columns to its right. The known 0-agents to the right of c, along with e being a 0-agent, allow the algorithm to infer all missing edges to be unsuccessful. The originally removed 1-agents had been matched to a total of $8i - 4$ 0-agents, and b and c had each been matched to $4i - 2$ 0-agents. In addition, $z = 8i - 4$, $w = 4$, and $x = 0$, so Inequality (3) is satisfied.

Otherwise, e is a 1-agent. In this case, the algorithm removes e's column, along with the $i-1$ columns to its left and the i columns to its right. Because the $i-1$ columns to the right of c, as well as the agent f, are known to be 0-agents, this allows the algorithm to infer all missing edges to catch up with a sub-factorization. The originally removed 1-agents had been matched to a total of $8i-4$ 0-agents, b was matched to $4i-2$ and e to $4i-1$. In addition, $z = 8i-4$, $w = 4$, and $x = 1$ (because c had been matched with b), so Inequality (3) is again satisfied.

6. Discovery is made during phase $\frac{m}{2}$ (m even):

   In this case, $k = 2$. Since no 1-agent is ever paired to the same 0-agent multiple times, the total regret is at most $\frac{1}{2} \cdot k \cdot (n - k) = n - k = n - k + \left\lfloor \frac{k}{4} \right\rfloor$.

We have accounted for all possible scenarios where the first 1-edge is discovered. Therefore, we have completed the proof of Lemma 17. ■

C A Weakest Link Instance with Regret Exceeding $n - k$

We consider the instance of the weakest link (AND) model with $n = 10$ agents, $k = 4$ of whom have high skill. We argue that any algorithm on this instance can be made to incur regret 7. Note that this is strictly more regret than the $n - k = 6$ guaranteed by Theorem 6. It also exactly matches the upper regret bound from Theorem 7. Hence, the Ring Factorization with Repairs Algorithm is optimal for this instance.

Regardless of the actions of the algorithm in rounds 1–2, the adversary will reveal no $(1, 1)$-teams. This is always possible since the exploration graph after round 2 is bipartite, so it includes an independent set of size 5. Next, we describe the actions of the adversary in round 3. To do this, we consider the possible exploration graphs $G(V, E_3)$. By a computer search, we determined that there are 102 non-isomorphic exploration graphs.

97 of these graphs have two (not necessarily disjoint) independent sets I_1, I_2 with $|I_1| = |I_2| = 4$ and such that $|I_1 \cup I_2|$ is odd. The adversary can use I_1 and I_2 to force the algorithm to incur regret at least 7. First, the existence of I_1 and I_2 allows the adversary to again reveal no $(1, 1)$-pairs in round 3. At the end of round 3, the algorithm has accrued 6 units of regret. Since $I_1 \cup I_2$ has odd cardinality, any matching that the algorithm chooses must include an edge connecting some agent in $I_1 \cup I_2$ to an agent $w \notin I_1 \cup I_2$. However, this gives the adversary a labeling that includes a $(0, 1)$-team in round 4, by making either all of I_1 or all of I_2 have type 1, depending on which of the two has the edge to w. This $(0, 1)$-edge adds at least one additional unit of regret. We draw these 97 graphs below, using red and blue dots to depict I_1 and I_2.

In the remaining 5 graphs (the last 5 graphs shown below), either (i) there is no independent set of size 4 or (ii) there is a matching such that for every independent set of size 4, each edge in the matching either connects two agents in the independent set or two agents not in the independent set. In either case, the adversary must reveal a $(1, 1)$-team in round 3, either to ensure a consistent type assignment or to force total regret 7.

For each of these graphs, we programmatically verified the following:

1. Given any three matchings whose union is the depicted graph, each of the three matchings includes one of the blue edges (or the image of one of the bolded blue edges under an automorphism of the graph).

2. By revealing the blue edge contained in the round-3 matching as a $(1, 1)$-edge, the adversary can always either (i) ensure regret 2 in round 4 (so the regret in round 3 is one, and in round 4, it is two) or (ii) ensure regret 1 in round 4 and leave two independent 2-sets I_1, I_2 in the unresolved subgraph such that $|I_1 \cup I_2| = 3$ (so the regret in each of rounds 3,4, and 5 is one).

Thus, the total regret in each of the cases is at least 7.