Pure Exploration and Regret Minimization in Matching Bandits

Flore Sentenac [*1] **Jialin Yi** [*2] **Clément Calauzènes** [3] **Vianney Perchet** [4] **Milan Vojnović** [2]

Abstract

Finding an optimal matching in a weighted graph is a standard combinatorial problem. We consider its semi-bandit version where either a pair or a full matching is sampled sequentially. We prove that it is possible to leverage a rank-1 assumption on the adjacency matrix to reduce the sample complexity and the regret of off-the-shelf algorithms up to reaching a linear dependency in the number of vertices (up to $\mathrm{poly}\log$ terms).

1. Introduction

Finding *matchings* in graphs, i.e., subsets of edges without common vertices, is a long standing problem (Lovász & Plummer, 2009) with many different applications in economics (Roth et al., 2004), operations research (Wheaton, 1990), and machine learning (Mehta, 2012). We consider here its sequential variant where at each time step t, an agent chooses a matching m_t of some graph, defined by its unknown weighted adjacency matrix W (with bounded elements in $(0,1)$), and observes noisy evaluations of the chosen entries $\{X_{i,j,t} : (i,j) \in m_t\}$, with $\mathbb{E}[X_{i,j,t}] = W_{i,j}$. This problem obviously falls in the realm of combinatorial bandits (Cesa-Bianchi & Lugosi, 2012), but we aim at leveraging a specific structural property: in many relevant examples, W is a rank 1 matrix.

Two different types of graphs are relevant for matchings, *bipartite* and *monopartite*, and we are going to consider both of them (even though the latter is more intriguing, the former, maybe more intuitive, will serve as a warm-up and to convey insights). In the bipartite case, the set of vertices is separated in two distinct subsets $\mathcal{U}$ and $\mathcal{V}$ (of respective sizes N and M) and edges only exist across subsets, not within. The rank-1 adjacency matrix W is then a $N \times M$ matrix, that can be written as $W = uv^\top$ for some $u \in (0,1)^N$ and $v \in (0,1)^M$. The canonical application of this setting is online advertising, where the probability that a user clicks on an ad depends on both the position at which the ad is displayed and its relevance to the user (Katariya et al., 2017b). Other motivations come from two-sided markets, where matching occurs between offers and demands, e.g. in online labor markets, u_i may represent the utility for a user seeking a solution to a project and v_j may represent the expertise of a project solver.

On the other hand, monopartite graphs have $2N$ vertices and their rank-1 adjacency matrix W can be written as $W = uu^\top$ for some $u \in (0,1)^{2N}$. This setting models collaborative activities that arise in teamwork, online gaming, and online labor platforms (Johari et al., 2018). For instance, some online gaming apps match players together (e.g. Go, competitive quizzes, and drawings) but players will participate only if they both want to. In a simple model, player i decides to participate with probability u_i, and matched players i and j participate in a game with probability $u_i u_j$. The expected number of played games from a proposed matching m is then $\sum_{(i,j)\in m} u_i u_j$. The revenue of such apps typically comes from ads displayed during games, so the more games played the better. In these examples, the app will match (say, everyday) as many pairs of players as possible and not just one (as in the bipartite example).

We will consider the aforementioned two variants of these sequential choices of matchings: either the matching has to be "minimal", i.e., it has to be a single pair of vertices, or it has to be "maximal", i.e., a choice of N distinct pairs. We will refer the former to as *pair selection* and the latter as *matching selection* problem. As standard in multi-armed bandits, we shall investigate both the regret minimization over an arbitrary given time horizon and the pure exploration in a PAC learning setting.

1.1. Related work

The matching problems defined above are special classes of combinatorial bandit problems with semi-bandit feedback (Cesa-Bianchi & Lugosi, 2012) with many recent improvements for regret minimization (Combes et al., 2015; Cuvelier et al., 2021; Degenne & Perchet, 2016; Perrault et al., 2020; Wang & Chen, 2021) as well as pure exploration (Garivier & Kaufmann, 2016; Chen et al., 2014). The combinatorial structure is quite clear, as the cardinality of the set of matchings is equal to $(2N)!/(2^N N!) \sim \sqrt{2}\big(2N/e\big)^N$.

[*]Equal contribution [1]CREST, ENSAE Paris, Palaiseau, France [2]London School of Economics, London, UK [3]Criteo AI Lab, Paris, France [4]CREST, ENSAE Paris, Palaiseau, France & Criteo AI Lab, Paris, France. Correspondence to: Flore Sentenac <flore.sentenac@gmail.com>, Jialin Yi <J.Yi8@lse.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Off-the-shelf combinatorial bandits algorithms would incur a regret scaling as $\tilde{O}(N^2 \log(T)/\Delta_{\min})$, where $\Delta_{\min}$ denotes the expected reward gap between an optimal matching and the best sub-optimal matching. This has been recently improved, but only in the aforementioned bipartite case, where the rank-1 structure has been leveraged in the line of work of *stochastic rank-1 bandits* (Katariya et al., 2017b;a; Trinh et al., 2020), yet either with sub-optimal parameter dependencies or with asymptotic performances. This quadratic dependency would also appear in pure exploration, as standard algorithms would require in the bipartite case $O(NM \log(1/\delta))$ iterations to find the best pair with probability at least $1-\delta$ (Garivier & Kaufmann, 2016).

The classical matching problem has strong connections with ranking/sorting; it is obviously the same with their sequential variants (Zoghi et al., 2017; Rejwan & Mansour, 2020) even though they do not directly handle the bandit feedback.

1.2. Organization of the paper and our contributions

The remaining of the paper is divided in four main parts. First, we formally introduce the general model in Section 2. Then we investigate the pair selection problem (both for regret minimization and pure exploration) in Section 3, and afterwards the matching selection problem (again, for regret and pure exploration) in Section 4. Finally, we present numerical results in Section 5. They validate the tightness of our results and demonstrate competitiveness and performance gains obtained by our proposed algorithms over some state-of-the-art baseline algorithms.

Our contributions can be summarized as follows:

i) For the pair selection problem in the bipartite case, we introduce a new algorithm, called PAIR-ELIM, in Section 3.1 with an optimal (up to a multiplicative constant) regret bound: perhaps interestingly, the algorithm eliminates sub-optimal rows and columns on different timescales. This result is extended to the monopartite case in the same Section 3.1.

   For pure exploration, we simply adapt PAIR-ELIM; it still leverages the rank-1 structure to find the optimal pair with a linear (instead of quadratic) sampling complexity in $O((N+M)\log(1/\delta))$.

ii) The monopartite case, still with pair sampling, is investigated in Section 3.2; we transform the above algorithm into PAIR-ELIM-MONO, that can handle both regret minimization and pure exploration. It is also extended for best matching identification, with again optimal bound (up to a multiplicative constant).

iii) Section 4.1 is dedicated to pure exploration with matching sampling; a new algorithm is developed with optimal sample complexity for non-degenerate ranges of parameters (i.e., it equals the new lower bounds proved up to multiplicative constants).

iv) Finally, regret minimization in the matching selection problem is investigated in Section 4.2; we introduce a new ADAPTIVE-MATCHING algorithm with a linear (instead of quadratic) dependency in N since its regret scales as $\tilde{O}(N \log(T)/\Delta_{\min})$.

   Roughly speaking, this algorithm relies on a divide and conquer type of approach.

2. Objectives and problem statement

Noise model. We assume the noisy observation X_t of W is generated as follow: for any (i,j,t), $X_{i,j,t} = W_{i,j} + \varepsilon_{i,j,t}$ where $\varepsilon_{i,j,t}$ are independent, zero-mean, sub-Gaussian random variables.

Optimal matching. The objective is to find a matching m, either minimal or maximal depending on the setting, that maximizes the expected reward $\mathbb{E}[\sum_{(i,j)\in m} X_{i,j,t}] = \sum_{(i,j)\in m} W_{i,j}$. It turns out that in both the bipartite case and the monopartite one, under the rank-1 assumption, the optimal matching is the one that pairs better items together. More formally and without loss of generality, for the bipartite case ($W = uv^\top$), we assume that $u_1 \geq \cdots \geq u_N$ and $v_1 \geq \cdots \geq v_M$. The optimal matching is the one that associates (u_1, v_1), then (u_2, v_2), and so on[1]. Similarly, for the monopartite case ($W = uu^\top$), we assume that $u_1 \geq u_1 \geq \cdots \geq u_{2N}$ and the optimal matching associates any odd index with its successor, i.e. (u_1, u_2) then (u_3, u_4) and so on. In both cases finding the optimal matching boils down to finding the order of the entries of u and v.

Pure exploration. A first objective the agent can aim for is to identify the best matching with *high probability* and *as fast as possible*. Formally, given a confidence level $0 < \delta < 1/2$, the agent seeks to minimize the worst-case number of samples τ_δ needed for the algorithm to finish and return the optimal matching with probability at least $1-\delta$.

Regret minimization. Another objective for the agent is to find the best matching while *playing sub-optimally as few times as possible* in the process. Formally, her goal is to minimize the regret, i.e., the difference between the cumulative reward of the oracle (that knows the best pair or the best matching) and her cumulative reward. Denoting by $\mathcal{M}$ the set of matching considered – e.g. minimal matchings for *pair selection* or maximal matchings for *matching selection* – the regret after T steps is defined as:

$$R(T) = T \max_{m\in\mathcal{M}} \sum_{(i,j)\in m} W_{i,j} - \sum_{t=1}^{T} \sum_{(i,j)\in m_t} W_{i,j}\,. \quad (1)$$

[1] This is a direct consequence of the *rearrangement inequality*.

Universal vs. parameter dependent constants. In order to avoid cumbersomeness, we shall use the notations $c_{\mathfrak{u}}$ to denote some universal constant and $c_{\mathfrak{p}}$ to denote constants (w.r.t. T) but that can depend on other problem parameters. They might change from one statement to another, but they are always defined explicitly in the proofs.

3. Pair selection problem

In this section we consider the pair selection problem. Even though playing one pair (i, j) of items at each time step may seem very similar to dueling bandits (Yue et al., 2012) in the monopartite case, the reward information structure is very different. In dueling bandits the information is *competitive*, one observes which i or j is best (in expectation). Here, the information is *collaborative*, the higher the parameters of both i and j, the higher the observation (in expectation). Thus, in our case, playing the pair (i, j) does not provide information on the relative order of i and j. Instead, to get information about the relative order of i and j, it is necessary to use a third item j' as a point of comparison and play both (i, j') and (j, j'). We will refer to this as comparing i with j *against* j'. A crucial idea, that is key to several of the algorithms presented in the paper, is that the fastest way to compare two items i and j is to compare them against the item j' with the highest possible parameter value. It turns out this last remark also holds in the bipartite case.

3.1. Bipartite case

As the bipartite case has already been studied and might be simpler to grasp, we start with it and then extend the results to the monopartite case. The fastest way to compare row items is to compare them against the best column, and reciprocally for columns. Similarly to RANK1ELIM (Katariya et al., 2017b), the algorithm maintains a list of *active* rows (resp. columns) that are, with high probability, *non-provably dominated* as defined by confidence sets computed from the samples. As RANK1ELIM, PAIR-ELIM performs an Explore Then Commit (ETC) strategy, playing all active rows against randomly chosen active columns to collect samples and update the confidence sets. Then it uses a similar ETC strategy on columns. The main difference with RANK1ELIM resides in PAIR-ELIM eliminating row and columns at different timescales. PAIR-ELIM implements an ETC policy with horizon T for rows. Simultaneously, it runs an ETC policy for columns, but over shorter time windows w (referred to as "blocks") between steps T_{w-1} and $T_w - 1$ where $T_w := T_{w-1} + 2^{2^w}$. Within a block, columns are only *temporarily* eliminated. They are reinstated as active at the beginning of the next block, when a new instance of the ETC is run in the new horizon. The detailed pseudo-code is given in Appendix B.

The key intuition is that the algorithm is more aggressive in the elimination of columns (i.e. with lower confidence) as they are only temporarily eliminated. Thus, sub-optimal rows will be eliminated after $\log(T)$ samples while the number of samples of sub-optimal columns only increases logarithmically in the current number of samples. This implies that pairs (i, j) that are "doubly-suboptimal", i.e. both i and j are sub-optimal, are eliminated after $\log(\log(T))$ samples. On the other hand, pairs that are only suboptimal, but not doubly, are eliminated after $\log(T)$ samples.

Algorithm Sketch 1 PAIR-ELIM

```
Set target precision to δ
for t = 0... do
  Identify active rows and columns
  Sample all active columns against a random active row
  Sample all active rows against a random active column
  if t > Σ_{s=0}^{w} 2^{2^s} then
    Reset samples for columns
    w = w + 1
  end if
  Update confidence sets on rows and columns
  if Optimal pair detected with confidence ≥ 1 − δ then
    Output optimal pair
  end if
end for
```

It is noteworthy that this cannot be achieved with a standard Explore Then Commit (ETC) independent on rows and columns, as the number of samples of sub-optimal decisions would then scale linearly (not logarithmically) with the number of samples – before elimination. This is the reason why the algorithm RANK1ELIM is sub-optimal.

The complexity of bandit problems is characterized by the *gaps*, i.e., the difference in expectation between basic item performances. In this case, we need to differentiate gaps on rows and columns, namely, $\Delta_i^U = \max_{i' \in [N]} u_{i'} - u_i$ and $\Delta_j^V = \max_{j' \in [M]} v_{j'} - v_j$ that appear both in the regret and in the sample complexity.

Theorem 3.1 *For any time horizon $T > 0$, the expected regret of* PAIR-ELIM *with $\delta = (1/T)$ is upper bounded as,*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} A(u, v) \log(T) + c_{\mathfrak{p}} \log(\log(T))$$

where

$$A(u, v) = \sum_{i \in [N] : \Delta_i^U > 0} \frac{1}{v_1 \Delta_i^U} + \sum_{j \in [M] : \Delta_j^V > 0} \frac{1}{u_1 \Delta_j^V}.$$

The proof of Theorem 3.1 is available in Appendix B.

Theorem 3.2 *For any $\delta \in (0, 1)$,* PAIR-ELIM *outputs the best pair with probability at least $1 - \delta$ at stage τ_δ s.t.*

$$\mathbb{E}[\tau_\delta] \leq c_{\mathfrak{u}} A(u, v) \log(1/\delta) + c_{\mathfrak{p}} \log\log(1/\delta)$$

where

$$A(u,v) = \sum_{i\in[N]:\Delta_i^U>0} \frac{1}{(v_1\Delta_i^U)^2} + \sum_{j\in[M]:\Delta_j^V>0} \frac{1}{(u_1\Delta_j^V)^2}$$

which is tight up to a multiplicative constant.

Proofs of the upper bound of Theorem 3.2 is in Appendix B. The lower was proven in previous work (Katariya et al., 2017b).

The improvement compared to RANK1ELIM is visible, as the leading term of the regret scales as the inverse of the parameters mean for RANK1ELIM, while it only scales as the inverse of the best parameter, for PAIR-ELIM.

3.2. Monopartite case

We introduce a new algorithm, PAIR-ELIM-MONO, that generalizes the main ideas of PAIR-ELIM. Instead of working on monopartite graph (of size $2N$), it first duplicates items and create a bipartite graph with $\mathcal{U} = \mathcal{V} = [2N]$. Then, as in the previous section, an elimination policy is run over rows with horizon T and over columns by blocks.

The major difference between the mono and bipartite case is that, in the former, two items of $\mathcal{U}$ and $\mathcal{V}$ are optimal (instead of only one, because of the initial duplication). As a consequence, active pairs are tracked instead of active rows and columns. Pairs containing item i are all eliminated after they have been deemed smaller than two other distinct items. If i is deemed smaller than another item j, all pairs containing item i are eliminated except (i,j). If entry (i,j) is eliminated as a consequence of row i's sub-optimality, entry (j,i) is also eliminated.

Note that the fastest way to compare item 1 with item $i > 2$ is to compare them against item 2 and the fastest way to compare item 2 with item $i > 2$ is to compare them against item 1. Similarly, it is harder to identify the second best item than the best item, as simple computations yield

$$u_1\Delta_{2,i} \leq u_2\Delta_{1,i}.$$

where $\Delta_{i,j} = u_i - u_j$.

Apart from the algorithm itself, another difference with the bipartite case is the way to measure the gaps, but the guarantees are very similar to the bipartite case.

Theorem 3.3 *The expected regret of* PAIR-ELIM-MONO *satisfies, for any time horizon $T > 0$,*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} A(u)\log(T) + c_{\mathfrak{p}}\log\log(T) \qquad (2)$$

where

$$A(u) = \sum_{i\in\{3,\dots,2N\}:\Delta_{2,i}>0} \frac{1}{u_1\Delta_{2,i}}.$$

The proof is provided in Appendix C.

Algorithm Sketch 2 PAIR-ELIM-MONO

```
Set target precision to δ
Initiate rows U = [2N] and columns V = [2N]
for t = 0, 1, ... do
  Identify active pairs
  Sample all active pairs
  if t > Σ_{s=0}^{w} 2^{2^s} then
    Reset samples for columns
    w = w + 1
  end if
  Update confidence sets on rows and columns
  if Optimal pair detected with confidence ≥ 1 − δ then
    Output optimal pair
  end if
end for
```

Theorem 3.4 *For any $\delta \in (0,1)$, the sample complexity of the* PAIR-ELIM-MONO *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} A(u)\log(1/\delta) + c_{\mathfrak{p}}\log\log(1/\delta)$$

with probability at least $1-\delta$, where

$$A(u) = \sum_{i\in[2N]:\Delta_i>0} \frac{1}{(u_1\Delta_{2,i})^2}$$

which is tight up to a multiplicative constant.

The proof is provided in Appendix C.

Towards maximal matchings. The *matching selection* setting can be seen as a constrained versions of *pair selection* where the agent has the constraint that N consecutively sampled pairs should form a maximal matching instead of being chosen freely. Hence, before diving in this setting, we can wonder what would be the sample complexity to identify the best maximal matching, but by freely choosing the pairs, which is arguably a simpler problem than *matching selection*. This can be done in two steps: **1)** identify the two best items using PAIR-ELIM-MONO, **2)** sample all unranked items against them until the full best matching is identified. We refer to this two-step algorithm as PAIR-SELECT. There again, the sample complexity of the algorithm is optimal up to a multiplicative constant, proofs are deferred to Appendix D.

Theorem 3.5 *For any $\delta \in (0,1)$, the sample complexity of the* PAIR-SELECT *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} A(u)\log(1/\delta) + c_{\mathfrak{p}}\log\log(1/\delta)$$

with probability at least $1-\delta$, where

$$A(u) = \sum_{i\in[2N]:\Delta_i>0} \frac{1}{(u_1\Delta_i)^2},$$

with $\Delta_{2,i} = u_{2i} - u_{2i+1}$ *and* $\Delta_{2i-1} = u_{2i-2} - u_{2i-1}$

which is tight up to a multiplicative constant.

Surprisingly, identifying the full order of the items rather than simply the top two ones does not require N times more samples. Rather the degradation appears in the gap, that are not anymore the gap to the second-best item, but rather the gap between consecutive matched pairs.

In the following section, we investigate if similar guarantees hold in the *matching selection* setting or if the constraints on the choice of consecutive pairs have a stronger impact.

4. Matching selection problem

In this section, we present our results for the matching selection problem, where recall at each iteration step a maximal matching of items needs to be selected. We first consider the objective of pure exploration, and then consider the regret minimization objective. In this section, pair i refers to the i^{th} pair of consecutive items (u_{2i-1}, u_{2i}), the pair with the i^{th} highest expected reward in the optimal matching.

Contrarily to the pair selection setting, where the same algorithm has tight guarantees for both regret minimization and pure exploration, the matching selection setting requires different algorithms. This can be understood from a simple example where the smallest gap between consecutive items $i, i+1$ appears at the bottom of the ranking (between low quality items). From a pure exploration point of view, the fastest way to rank them is to compare them against the top ranked item. However, this is sub-optimal from a regret point of view as misranking i and $i+1$ incurs a low regret, while playing $(1, i)$ or $(1, i+1)$ incurs a high regret. The pure exploration objective and the regret minimization objective are treated separately in the following as they require the use of different algorithms.

4.1. Pure exploration for matching

The algorithm for matching selection with an objective of pure exploration, referred to as MATCHING-ID, is based on this idea of comparing items with the top ones to rank them quickly (remember finding an optimal matching amounts to finding an order over the items). It makes use of two non-exclusive sets of items: $\mathcal{S} \subseteq [2N]$ is the set of unranked items (their exact rank is unknown) and $\mathcal{B}$ the set of items that are still potentially amongst the $|\mathcal{S}|$ best items. The algorithm proceeds through iterations, in each iteration sampling matchings from $\mathcal{B} \cup \mathcal{S}$ such that all distinct pairs of items are sampled once. Items from $\mathcal{S}$ are ranked by using the samples collected from matches with items in $\mathcal{B}$. This procedure continues until the rank of all items is known.

We introduce the following notations

$$\mu_{[2N]\setminus\{2k,2k+1\}} = \frac{1}{2N} \sum_{i \in [2N]\setminus\{2k,2k+1\}} u_i.$$

Algorithm Sketch 3 MATCHING-ID

$\mathcal{S} = [2N]$
while $\mathcal{S} \neq \emptyset$ **do**
  Compute the set $\mathcal{S}$ of unranked items
  Compute the set $\mathcal{B}$ of candidate $|\mathcal{S}|$ best items
  Sample each item in $\mathcal{B} \cup \mathcal{S}$ once against each other item in that set
  Update confidence intervals for the items in $\mathcal{S}$ using observed outcomes of matches with items in $\mathcal{B}$
end while

Let s and h denote the indices of the smallest and the second smallest gap, i.e. $s = \arg\min_{k \in [2,N-1]} \Delta_{2k,2k+1}$ and $h = \arg\min_{k \in [N-1]\setminus\{s\}} \Delta_{2k,2k+1}\mu_{[2N]\setminus\{2k,2k+1\}}$. We also define

$$\gamma_{\min} = \min_{k \in [N-1]} \{\mu_{[2N]\setminus\{2k,2k+1\}}\Delta_{2k,2k+1}\}.$$

To simplify the exposition of the result, we assume that the smallest gap is not between the two best pairs (general version of the result is given in Appendix H).

Theorem 4.1 *[upper bound] For any $\delta > 0$, the sample complexity of the* MATCHING-ID *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{\gamma_{\min}^2} \log(1/\delta) + c_{\mathfrak{p}}$$

with probability at least $1-\delta$. Moreover, by denoting,

$$\alpha := \min\left\{\frac{1}{2}\frac{(u_1+u_2)\Delta_{2s,2s+1}}{\mu_{[2N]\setminus\{2h,2h+1\}}\Delta_{2h,2h+1}}, 1\right\},$$

the following holds with probability at least $1-\delta$

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{(1-\alpha)^2(u_1^2+u_2^2)\Delta_{2s,2s+1}^2} \log(1/\delta) + c_{\mathfrak{p}}.$$

The proof of these upper-bounds is deferred to Appendix H. These upper bounds are tight, up to multiplicative factors, for some interesting regimes of parameters, as stated below.

Theorem 4.2 *Assume that stochastic rewards of item pairs have Gaussian distribution with unit variance. Then, for any δ-PAC algorithm, we have*

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}} \sum_{i \in [2N]:\Delta_i > 0} \frac{1}{\sum_{j=1}^{2N} u_j^2} \frac{1}{\Delta_i^2} \log(1/\delta)$$

and

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}} \frac{1}{u_1^2+u_2^2} \frac{1}{\Delta_{2s,2s+1}^2} \log(1/\delta).$$

In particular, if all gaps are equal, the first lower bound of Theorem 4.2 matches the first upper bound of Theorem 4.1 up to a multiplicative constant. On the other hand, in the

opposite regime where one gap is substantively smaller than all others, then it is the second bounds that are equivalent.

These results with matching matching should be put into perspective with their pendant, Theorem 3.5, for pair selection. In this case, the sample complexity of the latter is N times bigger than the one of the former. This might seem surprising at first sight as pair selection is an "easier" problem (without constraints). The reason is that with matching selection, the algorithm gets to observe N pairs at each iteration (and not just one). As a consequence, the overall number of pairs evaluation $X_{i,j,s}$ are actually of the same order.

This is quite surprising as selection matchings is much more constrained than selecting batches of N arbitrary pairs (possibly with repetitions and/or single items sampled more than just once in a batch). The main consequence is that the matching identification problem is *as difficult* with minimal than maximal matchings selection (and therefore in any intermediate case).

4.2. Regret minimization

We first describe and analyze a simplified matching divide and conquer algorithm, which is correct for problem instances satisfying a condition on model parameters introduced shortly. This simplified algorithm allows us to convey the key idea that underlies the design of a more complicated algorithm. Having described this simplified algorithm and shown the regret upper bound, we will remove the aforementioned condition and show a regret upper bound that holds for a matching divide and conquer algorithm.

4.2.1. Simple Adaptive Matching

We consider problem instances under the following assumption on model parameters:

Assumption 1 *The model parameters $u_1, \dots, u_{2N}$ are assumed to satisfy $u_{2i-1} = u_{2i}$, for all $i \in [N]$.*

Under Assumption 1, there exists an optimal matching such that every pair of matched items have equal parameter values. This assumption avoids some complications that arise due to uneven clusters in the divide and conquer procedure.

The Simple Adaptive Matching (SIMPLE-ADAPTIVE-MATCHING) successively partitions items into an ordered sequence of clusters (ranked clusters), such that all items in a cluster have a higher rank than all items in any lower-ranked cluster with a high probability. Items in a cluster $\mathcal{S}$ of size $|\mathcal{S}| = 2K$ are matched according to a round-robin tournament, which runs over $2K - 1$ iterations.

A cluster is split into two sub-clusters as soon as the upper bound for the reward of each item in one of the sub-clusters is smaller than the lower bound for the reward of each item in the other sub-cluster. Assumption 1 guarantees that, with high probability, at each iteration step, all clusters contain an *even* number of items, so it is always possible to match all items within each cluster.

Functions `sample_matching` and `conf_bound` of Algorithm 4 are detailed in Appendix E. At the high level, `sample_matching` samples matchings that ensure that each $|\mathcal{S}|-1$ iterations, any given item in $\mathcal{S}$ has been matched once with any other item in $\mathcal{S}$. `conf_bound` builds confidence intervals for the total reward of each item per match with items in the same cluster or in lower ranked cluster.

Algorithm 4 SIMPLE-ADAPTIVE-MATCHING

Input: set of items $[2N]$ and horizon T
$t = 0, C = X = \tilde{C} = \tilde{X} = [0]^{2N \times 2N}, \mathfrak{S} = \{[2N]\}$
for $t = 1 \dots T$ **do**
  $m_t \leftarrow$ `sample_matching`$(\mathfrak{S}, t)$
  for $(i, j) \in m_t$ **do**
    $\tilde{X}(i,j) \leftarrow \tilde{X}(i,j) + X_{i,j,t}$
    $\tilde{C}(i,j) \leftarrow \tilde{C}(i,j) + 1$
  end for
  for $\mathcal{S} \in \mathfrak{S}$ **do**
    if $\exists i \in \mathcal{S}$ s.t. $\sum_{j \in S} \tilde{C}(i,j) = |\mathcal{S}| - 1$ **then**
      $X([s],:), C([\mathcal{S}],:) += \tilde{X}([\mathcal{S}],:), \tilde{C}([\mathcal{S}],:)$
      $\tilde{X}([\mathcal{S}],:), \tilde{C}([\mathcal{S}],:) = 0$
    end if
  end for
  $Q_+, Q_- \leftarrow$ `conf_bound`$(X, C, T, Q_+, Q_-, \mathfrak{S})$
  for $\mathcal{S} \in \mathfrak{S}$ **do**
    Order items in $\mathcal{S}$ according to Q_+
    for $i \in \{2, \dots, |\mathcal{S}|\}$ **do**
      if $Q_+[i] < Q_-[i-1]$ **then**
        Split $\mathcal{S}$ between $i-1$ and i
      end if
    end for
  end for
end for

The regret of SIMPLE-ADAPTIVE-MATCHING can be bounded by using the following additional notation

$$\Delta_{\min} = \min_{m \in \mathcal{M}: m \neq m^*} \Big\{ \sum_{(i,j) \in m^*} u_i u_j - \sum_{(i,j) \in m} u_i u_j \Big\}$$

and the proof is again deferred to Appendix E.

Theorem 4.3 *The expected regret of* SIMPLE-ADAPTIVE-MATCHING *satisfies, for any horizon $T > 0$,*

$$\mathbb{E}[R(T)] \leq c_{\mathtt{u}} \frac{N \log(N)}{\Delta_{\min}} \log(T) + c_{\mathtt{p}}.$$

4.2.2. Adaptive-Matching algorithm

In general, when Assumption 1 does not hold, some of the clusters may have odd sizes. In these odd-size clusters, uniform sampling within a cluster is infeasible, and we need

to match items residing in different clusters. In order to deal with these complications, we use a new ADAPTIVE-MATCHING algorithm.

ADAPTIVE-MATCHING is defined as an extension of SIMPLE-ADAPTIVE-MATCHING. It uses the same policy for splitting clusters. The SAMPLE-MATCHING procedure extends that of the previous algorithm to unevenly split clusters. This procedure ensures that any two mutually unranked items are matched similarly to other items, which guarantees that the expected rewards for those items are scaled similarly. At the high level, it defines a list of matchings that respect the desired item sampling proportion, then samples the matchings in this list in a round-robin.

A detailed description of SAMPLE-MATCHING, and the proof of the following result, are given in Appendix F.

Theorem 4.4 *The expected regret of* ADAPTIVE-MATCHING *is bounded, for any horizon $T > 0$, as*

$$\mathbb{E}[R(T)] \leq c_{\mathrm{u}} \frac{N \log(N)}{\Delta_{\min}} \log(T) + c_{\mathrm{p}}.$$

4.2.3. COMPARISON WITH AN EXPLORATION POLICY

The ADAPTIVE-MATCHING algorithm matches high parameter value items together as soon as they are identified in order to exploit this for accruing reward. On the other hand, our algorithms, for the pair sampling problem and the pure exploration matching identification problem, used the detected high parameter value items to *explore* the unranked ones. Without a comparison, it is unclear which of the two strategies will lead to a smaller regret in the end. For this reason, we consider an *exploration-first* algorithm that matches identified high parameter value items with other items to speed up the learning of the rank of these other items, and compare it with ADAPTIVE-MATCHING.

We do this under assumption that both algorithms are given as input the two best items, as well as a set of $2(N-1)$ unranked items. We chose as a comparison metric the upper bound on the total regret incurred by the two algorithms until the un-ranked items can be partitioned into two or more ranked clusters of items. We denote with U_I the upper bound on the regret R_I for the exploration-first strategy and with U_D the upper bound on the regret R_D for the ADAPTIVE-MATCHING algorithm.

The following Lemma 4.5 states that unless the second best item is sufficiently worse than the best item, the regret of the ADAPTIVE-MATCHING algorithm is at most of the same order as that of the exploration-first algorithm, and can be arbitrarily smaller depending on the problem parameters. If the ratio between the parameter values of the first and the second best item goes to zero, then the exploration-first algorithm becomes infinitely better than the ADAPTIVE-MATCHING algorithm.

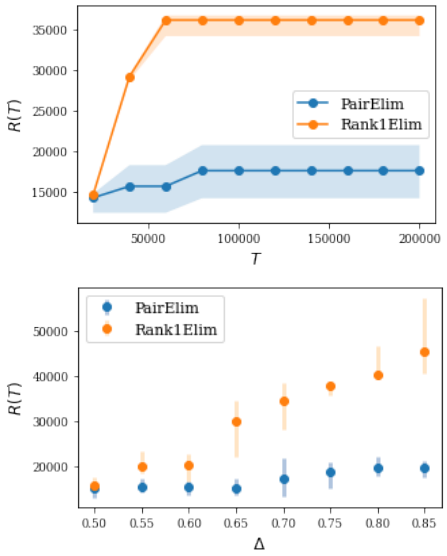

Figure 1: Regret comparison for PAIR-ELIM vs RANK1ELIM: (top) regret versus T for fixed $\Delta = 0.75$ and (bottom) regret versus the gap parameter Δ for fixed $T = 2,000,000$. We used 20 independent runs. The shaded areas show the range between the 5% and 95% percentile.

Lemma 4.5 *If $u_2/u_1 > 1/2$, then $U_D/U_I \leq c_{\mathrm{u}} N$, and it can be arbitrarily close to 0 depending on the problem parameters. If $u_3/u_2 > 1/2$, then this ratio is smaller than a constant independent from the parameters of the problem. On the other hand, $\lim_{u_2/u_1 \to 0} R_D/R_I = +\infty$.*

In summary, the lemma tells that ADAPTIVE-MATCHING is essentially as good as the exploration-first strategy for any problem instance such that the parameter value of the second best item is at least a constant factor of the best item.

5. Numerical results

In this section we present numerical results, which demonstrate tightness of our theoretical bounds and compare our proposed algorithms with some state-of-the-art baseline algorithms. We first consider the pair selection problem and then the matching selection problem. In summary, our numerical results validate our theoretical results and demonstrate that significant performance gains can be achieved against some previously proposed algorithms.

All the code used for obtaining the results in this section is available from this public Gitlab repository: [anonymized]

5.1. Pair selection

We consider RANK1ELIM as a baseline for comparison. As noted in the introduction, RANK1ELIM has a regret upper bound that is sub-optimal with respect to the problem parameters, which is in contrast to our algorithm, PAIR-ELIM

that has optimal regret bound up to a multiplicative constant. We demonstrate that significant performance gains that can be achieved by using PAIR-ELIM versus RANK1ELIM for some problem instances.

In all experiments, the variables considered are Bernoulli variables. We consider the bipartite case with $N = M$. Each problem instance is defined by a tuple (N, u_1, Δ), where $0 \leq \Delta \leq u_1 \leq 1$, and assuming that $u_1 = v_1$. The row parameter values $u_2, \ldots, u_N$ are defined as sorted values of independent random variables according to uniform distribution on $[0, 2(u_1 - \Delta)]$, where Δ is the expected gap between the value of the best item and the value of any other item. Note that we have $\mu_U = u_1 - (1 - 1/N)\Delta$. For fixed value of parameter u_1 and increasing expected gap Δ, we have problem instances with fixed maximum row parameter value and decreasing mean row parameter value μ_U. We similarly define the column parameter values, and all the observations above made for row parameter values hold for column parameter values. According to our regret analysis, PAIR-ELIM algorithm will outperform RANK1ELIM when Δ is large for fixed u_1. To confirm this claim, we ran the two algorithms on a set of problem instance with $N = 8$ and $u_1 = 0.9$. The results are shown in Figure 1. We have further evaluated the effects of varying u_1 which is discussed in Appendix A.1.

5.2. Matching selection

In this section we evaluate the performance of our algorithm for the matching selection problem. Our goal is twofold. We first demonstrate numerical results according to which our proposed algorithm has expected regret that scales proportionally to $N \log(N)/\Delta_{\min}$ for a fixed horizon T. We then compare its performance with that of ESCB, which, as discussed in the introduction, has the expected regret bound $O(N^2 \log^2(N)/\Delta_{\min})$ for fixed T. We demonstrate that our algorithm can achieve significant performance gains over ESCB. In our evaluations. We use our SIMPLE-ADAPTIVE-MATCHING algorithm, for problem instances such that there is a unique optimum matching with matched items having equal parameter values.

We first show that the regret of our proposed algorithm scales in the order of $N \log(N)/\Delta_{\min}$. We consider a set of problem instances, each is defined by a tuple $(N, \tilde{\Delta})$, where $0 < (N-1)\tilde{\Delta} \leq 1$ and $\tilde{\Delta}$ is the gap between parameter values of adjacent matched pairs in the optimum matching, so that $\tilde{\Delta} = \sqrt{\Delta_{\min}}$. The parameter values are defined by $u_{2i-1} = u_{2i} = (N-i)\tilde{\Delta}$ for $i \in [N]$. We run our algorithm on problem instances with $\tilde{\Delta} = 0.1$. The results shown in Figure 2 suggest that the cumulative regret of SIMPLE-ADAPTIVE-MATCHING scales as $(1/\Delta_{\min})N \log(N)$ as established in Theorem 4.3.

We next compare the performance of our algorithm and ESCB. We consider problem instances defined by a tuple

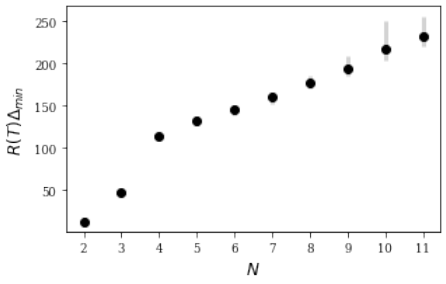

Figure 2: Normalized regret of SIMPLE-ADAPTIVE-MATCHING versus N for $u_{2i-1} = u_{2i} = (N-i)\tilde{\Delta}$, for $i \in [N]$, with $\tilde{\Delta} = 0.1$ and $T = 200,000$.

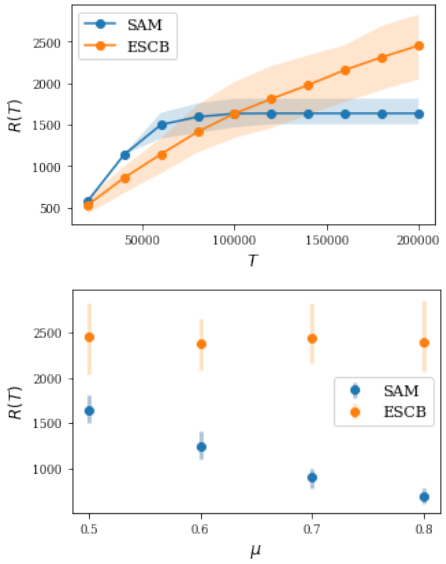

Figure 3: Regret comparison for SIMPLE-ADAPTIVE-MATCHING (SAM) vs ESCB: (top) regret versus T for fixed $\mu = 0.5$ and (bottom) regret vs μ for fixed $T = 200,000$.

$(N, \mu, \tilde{\Delta})$ where μ is the mean of parameter values and $\tilde{\Delta}$ is the gap between parameter values. The values of item parameters are set as $u_{2i-1} = u_{2i} = \mu + (N+1-2i)\tilde{\Delta}/2 \in [0,1]$ for $i \in [N]$. Such problem instances allow us to vary μ while keeping other parameters fixed; we fix $N = 4$ and $\tilde{\Delta} = 0.1$. In Figure 3 (top) we show the regret versus the time horizon T for fixed $\mu = 1/2$, which shows that our algorithm outperforms ESCB for large enough values of T. We expect our algorithm to perform better than ESCB as we increase μ. The results in Figure 3 (bottom) confirm this claim. We have performed these experiments for a small value of N because of the computation complexity of ESCB. ESCB requires solving an NP-hard problem in each iteration, and has overall computation complexity $O(|\mathcal{M}|T)$ where $\mathcal{M}$ is the set of all arms. For the matching selection problem, $|\mathcal{M}|$ scales as $\sqrt{2}(2N/e)^N$.

Acknowledgements

V. Perchet acknowledges support from the ANR under grant number ANR-19-CE23-0026 as well as the support grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences. M. Vojnović was supported in part by the Criteo Faculty Research Award.

References

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

Chen, S., Lin, T., King, I., Lyu, M. R., and Chen, W. Combinatorial pure exploration of multi-armed bandits. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 379–387. 2014.

Combes, R., Shahi, M. S. T. M., Proutiere, A., et al. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pp. 2116–2124, 2015.

Cuvelier, T., Combes, R., and Gourdin, E. Statistically efficient, polynomial time algorithms for combinatorial semi bandits, 2021.

Degenne, R. and Perchet, V. Combinatorial semi-bandit with known covariance. In *NIPS 2016 (Conference on Neural Information Processing Systems)*, 2016.

Garivier, A. and Kaufmann, E. Optimal best arm identification with fixed confidence. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 998–1027, Columbia University, New York, New York, USA, 23–26 Jun 2016.

Johari, R., Kamble, V., Krishnaswamy, A. K., and Li, H. Exploration vs. exploitation in team formation. *CoRR*, abs/1809.06937, 2018. URL http://arxiv.org/abs/1809.06937.

Katariya, S., Kveton, B., Szepesvári, C., Vernade, C., and Wen, Z. Bernoulli rank-1 bandits for click feedback. IJCAI'17, pp. 2001–2007. AAAI Press, 2017a.

Katariya, S., Kveton, B., Szepesvari, C., Vernade, C., and Wen, Z. Stochastic Rank-1 Bandits. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 392–401, Fort Lauderdale, FL, USA, 20–22 Apr 2017b.

Kaufmann, E., Cappé, O., and Garivier, A. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17 (1):1–42, 2016.

Lovász, L. and Plummer, M. D. *Matching theory*, volume 367. American Mathematical Soc., 2009.

Mehta, A. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012.

Perrault, P., Boursier, E., Perchet, V., and Valko, M. Statistical efficiency of thompson sampling for combinatorial semi-bandits. In *NeurIPS*, 2020.

Polyanskiy, Y. and Wu, Y. Lecture notes on information theory. *Lecture Notes for ECE563 (UIUC) and*, 6(2012-2016):7, 2014.

Rejwan, I. and Mansour, Y. Top-k combinatorial bandits with full-bandit feedback. In Kontorovich, A. and Neu, G. (eds.), *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pp. 752–776, San Diego, California, USA, 08 Feb–11 Feb 2020. PMLR.

Roth, A. E., Sönmez, T., and Ünver, M. U. Kidney exchange. *The Quarterly journal of economics*, 119(2):457–488, 2004.

Trinh, C., Kaufmann, E., Vernade, C., and Combes, R. Solving Bernoulli rank-one bandits with unimodal Thompson sampling. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pp. 862–889, San Diego, California, USA, 08 Feb–11 Feb 2020.

Wang, S. and Chen, W. Thompson sampling for combinatorial semi-bandits, 2021.

Wheaton, W. C. Vacancy, search, and prices in a housing market matching model. *Journal of political Economy*, 98(6):1270–1292, 1990.

Yue, Y., Broder, J., Kleinberg, R., and Joachims, T. The k-armed dueling bandits problem. *J. Comput. Syst. Sci.*, 78(5):1538–1556, September 2012. ISSN 0022-0000. doi: 10.1016/j.jcss.2011.12.028. URL https://doi.org/10.1016/j.jcss.2011.12.028.

Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., and Wen, Z. Online learning to rank in stochastic click models. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 4199–4208, International Convention Centre, Sydney, Australia, 06–11 Aug 2017.