Sequential Blocked Matching

Nicholas Bishop
University of Southampton, UK
nb8g13@soton.ac.uk

Hau Chan
University of Nebraska-Lincoln, USA
hchan3@unl.edu

Debmalya Mandal
Columbia University, USA
dm3557@columbia.edu

Long Tran-Thanh
University of Warwick, UK
long.tran-thanh@warwick.ac.uk

June 15, 2021

Abstract

We consider a sequential blocked matching (SBM) model where strategic agents repeatedly report ordinal preferences over a set of services to a central mechanism. The central mechanism's goal is to elicit agents' true preferences and design a policy that matches services to agents in order to maximize the expected social welfare with the added constraint that each matched service can be *blocked* or unavailable for a number of time periods. Naturally, SBM models the repeated allocation of reusable services to a set of agents where each allocated service becomes unavailable for a fixed duration.

We first consider the offline SBM setting, where the the strategic agents are aware of the true preferences. We measure the performance of any policy by *distortion*, the worst-case multiplicative approximation guaranteed by any policy. For the setting with S services, we establish lower bounds of $\Omega(S)$ and $\Omega(\sqrt{S})$ on the distortions of any deterministic and randomised mechanisms, respectively. We complement these results by providing approximately truthful, measured by *incentive ratio*, deterministic and randomised policies based on the repeated application of random serial dictatorship that match the lower bounds. Our results show that there is a significant improvement if one considers the class of randomised policies.

Finally, we consider the online SBM setting with bandit feedback where each agent is unaware of her true preference, and the center must facilitate each agent in the learning of agent's preference through the matching of services over time. We design an approximately truthful mechanism based on the Explore-then-Commit paradigm, which achieves logarithmic dynamic approximate regret.

1 Introduction

In recent years, machine learning algorithms have been extremely successful in various domains, from playing games to screening cancer. However, despite such success, most learning algorithms cannot be deployed directly in practice to make decisions under uncertainty. The main reason is that most real-world applications involve multiple agents, and learning algorithms are often constrained due to the unavailability of resources. In this paper, we consider the problem of repeated matching with blocking constraints, a scenario where multiple agents simultaneously learn their preferences with repeated blocking or unavailability of resources.

In particular, we are interested in the repeated *one-sided matching* problem where strategic agents report only their ordinal preferences based on their expected rewards over a set of alternatives.

The agents are matched to the alternatives given their reported preferences each time period or round. It is well-known that one-sided matching can be used to model various real-world situations such as matching patients to kidneys across health institutions [18, 43], assigning students to rooms in residential halls [18], allocating workers to tasks in crowdsourcing [17, 2], and recommending users to activities in recommender systems [45, 4, 24].

In many of these situations, there are two main obstacles. First, for a setting with reusable alternatives or *services*, a major caveat is that an agent-alternative match within a round can result in *blocking* of some services in which the services may not be available until a later time. For example, a recommended activity (e.g., a special promotion offer from a restaurant) that is matched to (or used by) a user may not be available to all users again until a later time. Or, in cloud computing, where tasks are matched to resources (e.g. GPUs), once a task is assigned to a resource, that resource is blocked for a certain number of rounds.

Second, the agents are often unaware of their exact preferences so that the central mechanism must coordinate their explorations without incurring a significant loss. This is often true for recommending restaurants to customers, as the restaurants have limited capacity and people are rarely informed of all possible choices. Furthermore, even if the agents are aware of their preferences, they might be strategic in reporting their preferences in order to be matched to better services. This is particularly prominent in assigning rooms to students. Rooms can be blocked due to occupancy and can be made available again once the students leave the rooms. As a result, there is a potential for the students to misreport their private preferences to manipulate the matching outcomes.

1.1 Main Contributions

To capture the notion of one-sided matching with blocking, we introduce a *sequential blocked matching* (SBM) model where a set of N strategic agents is matched to a set of S services repeatedly over rounds, and the match (i.e., the service) is *blocked* or unavailable for a deterministic number of rounds. Each agent strategically reports only the ordinal preference at each round, based on her expected rewards over the services and on the matches in previous rounds, to maximize her expected utility (i.e., the sum of the rewards of the matches) over all the rounds. The planner's goal is to design a central mechanism to derive a matching policy that, at each round, elicits true preferences from the agents and matches agents to services to maximize the expected social welfare, which is the sum of the expected utilities of the agents from the matching over rounds, while accounting for potential blocking from the matching. To the best of our knowledge, SBM models have not been studied before and can be applied to a wide range of real-world matching scenarios.

Table 1: Lower and Upper Bound Results for Offline SBM Models

	Distortion	Incentive Ratio
Any Deterministic Mechanism (*lower bound*)	$\Omega(S)$	$(0, 1]$
Derandomized Repeated Random Serial Dictatorship (*upper bound*)	$\mathcal{O}(S)$	$(1 - 1/e)$
Any Randomised Mechanisms (*lower bound*)	$\Omega(\sqrt{S})$	$(0, 1]$
Repeated Random Serial Dictatorship (*upper bound*)	$\mathcal{O}(\sqrt{S})$	$(1 - 1/e)$

We investigate the offline and online variations of the SBM model where agents can have preference and reward uncertainty over the set of services. Given the variations, we are interested in deriving deterministic and randomized mechanisms that are approximately (expected) truthful and efficient.

We measure truthfulness by *incentive ratio* [16], which measures how much a single agent can gain via misreporting preferences. We measure efficiency through the notion of *distortion* from social choice theory [41], which measures the loss in social welfare due to access to only preferences, and not utility functions and rewards. We formally define these concepts in Section 3.

Offline SBM Benchmarks. For the offline setting of SBM models, the agents know their preferences and rewards over the services, but the planner does not know about agents' true preferences. Essentially, the offline benchmarks establish what we can achieve in terms of distortion if the agents' don't have to learn. Table 1 summarizes our results. More specifically, we derive lower bounds on the distortion of any deterministic and randomised mechanism. The main ingredient of our proof is the careful construction of reward profiles that are consistent with reported preferences that guarantees poor social welfare for the planner. We then focus on the upper bound and provide approximately truthful mechanisms with bounded incentive ratios that match the distortion lower bounds. In short, both the deterministic and randomised mechanisms we provide are based on the repeated application of random serial dictatorship (RSD) mechanisms for one-shot one-sided matching problems. Our randomised mechanism, repeated RSD (`RRSD`), iterates randomly over all agents, greedily choosing the current agents' preferred service at each time step. Our deterministic mechanism, derandomised `RRSD` (`DRRSD`), is a derandomised version of this algorithm and matches the corresponding lower bound. Interestingly, we find that there is a strict separation of $\sqrt{S}$ between the achievable distortion by a deterministic and randomized mechanism.

Online SBM Algorithms. For the online setting of SBM, the agents do not know their preferences and rewards and must learn their preferences via repeated matching to services. After each matching, the agents update their preferences/rewards and strategically report them to the planner. We design an approximately truthful mechanism based on the Explore-then-Commit paradigm, which achieves sublinear dynamic approximate regret. In particular, we design a central allocation mechanism which consists of two phases. In the first phase, it allows the participating agents to use their favourite learning-to-rank with a fixed confidence algorithm. Using the learnt estimates from this phase, the mechanism then runs `RRSD` in the second phase. By doing so, we prove that the mechanism has a regret of $O(\log(NT))$, compared to the performance of `RRSD` in the offline setting.

1.2 Related Work

We provide a brief discussion of the related work in the matching and bandit literature and highlight major differences comparing to our SBM models, which have not been considered previously.

Blocking Bandits. Our work in the online SBM models is closely related to the recent literature on the blocking bandit models [9, 10, 12] where each pulled arm (i.e., services) can be blocked for a fixed number of rounds. Our work is also related to bandits with different types of arm-availability constraints [39, 34, 35]. However, these models do not consider the sequential matching setting where multiple strategic agents have (possibly unknown) ordinal preferences over arms and report ordinal preferences to the mechanism in order to be matched to some arms at each round.

Multi-agent multi-armed bandits. Our work is broadly related to the growing literature on multi-agent multi-armed bandits [37, 44, 13]. Liu et al. [37] consider a matching setting where strategic agents learn their preferences over time, and the center outputs a matching every round based on their reported preferences. However, our setting is more challenging as we need to compete against a dynamic offline benchmark because of the blocking of the arms, whereas the existing works compete against a fixed benchmark e.g. repeated applications of Gale-Shapley matching in each round [37].

Ordinal Matching and Distortion. We consider the objective of maximizing expected rewards as our offline benchmark. Since we do not observe the exact utilities of the agents rather ordinal preferences over items, we use the notion of *distortion* [41] from voting to quantify such a benchmark. In the context of voting, distortion measures the loss of performance due to limited availability of reward profiles [14, 38, 5, 32, 6]. Our offline benchmark is related to the literature on the distortion of matching [3, 19, 7]. However, our offline benchmark needs to consider repeated matching over time, and because of the blocking of the arms, has a very different distortion than the distortion of a single-round matching.

Online Matching. There are existing online notions of weighted bipartite matching (e.g., [28, 26, 27]) and stable matching (e.g., [33]) where the matching entities (e.g., agents or services) arrive dynamically over time and the corresponding information in the notions is publicly known (e.g., weights of the matched pairs or agents' ordinal preferences). These online settings are different from our repeated matching settings where the entities do not arrive dynamically and our objective is to maximize expected rewards of the repeated matching given agents' ordinal preferences. Other recent works explore dynamic preferences of the agents that can change over time (e.g., [40, 22, 23]). However, they do not consider the problem of maximizing expected rewards and matching blocking.

2 Preliminaries

We consider a sequential blocked matching (SBM) model which takes place over T rounds. In such a model, we have a set $\mathcal{K} = \{1, \ldots, N\}$ of N agents. For each agent, there is a set $\mathcal{S} = \{1, \ldots, S\}$ of S reusable *services* available to be matched to the agent. Furthermore, we assume that $S \gg N$. No generality is lost in making such an assumption, as we can always add services with zero rewards and no blocking. With each agent-service pair, $(i, j) \in \mathcal{K} \times \mathcal{S}$, we associate a mean reward $\mu_{i,j}$, which represents the *expected* reward agent i receives when allocated service j under some unknown distribution and is not known to the planner.

We denote by $\mu_i = (\mu_{i,1}, \ldots, \mu_{i,S})$ the vector of expected rewards associated with agent i. In what follows, we will also refer to μ_i as the *reward profile* associated with agent i. Moreover, we restrict ourselves to reward profiles which lie in the probability simplex. That is, we assume $\mu_i \in \Delta^S \ \forall i \in \mathcal{K}$. In other words, we make a unit-sum assumption about the reward profile of each agent. Bounding constraints on reward profiles are common in the ordinal one-sided matching literature [19], and are typically required in order to prove lower bounds for truthful algorithms such as random serial dictatorship (RSD). Moreover, the unit-sum assumption is prevalent in social choice theory [14].

At the start of each time step, each agent is required to submit ordinal preferences over services to a central mechanism. In other words, each agent i must submit a linear ordering $o_{i,t}$ over services to a central mechanism. We say that an agent submits its ordinal preferences truthfully if $a \succ_{o_{i,t}} b$ implies $\mu_{i,a} \geq \mu_{i,b}$ for all pairs of services a and b in $\mathcal{S}$.

The central mechanism is then tasked with matching agents to services based on the ordinal preferences submitted by each agent. At the end of each time step each agent i observes a reward $r_{i,t} \in [0, 1]$ dependent on the service it was matched to. If agent i is mapped to agent j then $r_{i,t}$ is sampled from a stationary distribution with mean $\mu_{i,j}$. The agents submit their ordinal preferences at each time step with the goal of maximising their cumulative rewards or utility over the time horizon.

Matchings. More formally, we define a *matching* as a function from $\mathcal{K}$ to $\mathcal{S} \cup \{0\}$, such that the restriction of any matching to the inverse image of $\mathcal{S}$ is an injection. If an agent i is mapped to zero, this implies the agent is allocated no services in the matching. We denote the set of all matchings by

$\mathcal{M}$. Given a matching m, we denote the service matched to agent i by $m(i)$. We denote the empty matching (the matching that does not match any service) by $\emptyset$.

We denote by $M = (m_t)_{i=1}^T$ a sequence of T matchings, where each matching m_t is associated with a time step t. We denote by $\mathcal{M}_T$ the set of all matching sequence of length T.

Blocking. Furthermore, we assume that when a service is matched, it is blocked for a time period depending on the agent it was matched to. More specifically, when agent i is matched with service j we assume that service j cannot be matched to any agent for the next $D_{i,j} - 1$ time steps. In what follows, we will refer to $D_{i,j}$ as the blocking delay associated with the agent-service pair i and j. Additionally, we let $\tilde{D}$ denote the maximal blocking delay possible and let D denote the N by S matrix of all blocking delays. From now on, we assume that all blocking delays are known apriori by both the central mechanism and the agents. We say that a matching sequence M is feasible with respect to the delay matrix D if no service is matched to an agent on a time step where it has been blocked by a previous matching.

Definition 1. *For a given blocking delay matrix D, the set of feasible matching sequences of length T, $\mathcal{M}_T^D \subseteq \mathcal{M}_T$, is the set of all matching sequences $M \in \mathcal{M}_T$ such that for all $t \in \{1, \dots, T\}$, for all $i \in \mathcal{K}$, and all $j \in \mathcal{S}$ the following implication holds:*

$$M(t,i) = j \implies M(t', \cdot) \neq j \quad \forall t' \text{ such that } t' > t \text{ and } t' \leq t + D_{i,j} - 1.$$

Note that blocking of services is a common phenomenon in real-world scenarios. For example, consider a setting in which each service corresponds to a freelance contractor, and each agent corresponds to an employer. The matching of services and agents then corresponds to employers contracting freelancers, For the duration of the contract, which may differ from employer to employer, the matched freelancer is unavailable before returning to the pool of available services once their contract ends.

Utilities and Welfare. We define the utility, $\mathrm{W}_i(M)$, agent i receives from a matching sequence M as the sum of rewards it receives from each matching in expectation. That is $\mathrm{W}_i(M) = \sum_{t=1}^T \mu_{i,m_t(i)}$. As previously mentioned, we assume that each agent submits ordinal preferences at each time step with the goal maximising her own utility. Moreover, we define the social welfare, $\mathrm{SW}(M)$, as the summation of the utilities of all agents. That is, $\mathrm{SW}(M) = \sum_{i=1}^N \mathrm{W}_i(M)$. The goal of the central mechanism is to produce a feasible matching sequence which maximises social welfare.

Matching Policies. Let U be a random variable over some probability space $(\mathbb{U}, \mathcal{U}, \mathbf{P}_u)$. Furthermore, let $\mathcal{O}$ denote the set of all linear preference orderings over services. Let, $\pi_1 : \mathbb{U} \times \mathcal{O}^N \times [\tilde{D}]^{N \times S} \to \mathcal{M}$ and $\pi_t : \mathbb{U} \times \mathcal{M}_{t-1} \times \mathcal{O}^{N \times t} \times [\tilde{D}]^{N \times S} \to \mathcal{M}$ for $t = 2, \dots$ be measurable functions. With some abuse of notation, we denote by $\pi_t \in \mathcal{M}$ the matching chosen at time t:

$$\pi_t = \begin{cases} \pi_1(U, H_1, D) & t = 1 \\ \pi_t(\pi_1, \dots, \pi_{t-1}, H_t, D, U) & t = 2, 3, \dots \end{cases}$$

where $H_t = (o_{1,1}, \dots\dots o_{N,1}, \dots, o_{1,t}, \dots o_{N,t})$ denotes the history of ordinal preferences submitted by all agents up to time step t. The mappings $\{\pi_t : t = 1, \dots, T\}$, together with the distribution $\mathbf{P}_u$, define the class of matching policies. We define the subclass $\mathcal{P}$ of admissible policies to be those which always return feasible matching sequences. That is,

$$\mathcal{P} = \left\{ \pi = (\pi_1, \dots, \pi_T) : \pi \in \mathcal{M}_T^D, \forall D \in [\tilde{D}]^{N \times S}, \forall H_T \in \mathcal{O}^{N \times T} \right\}.$$

Objective. For a given SBM instance, the goal of the mechanism is to find an admissible matching policy which maximises the social welfare in expectation with respect to randomisation of the policy, $\mathbb{E}_\pi[\mathrm{SW}(\pi)]$, under the assumption that agents will submit ordinal preferences with the intention of maximising their own utilities in expectation (again, with respect to any randomisation of the policy).

Settings. Of course, the difficulty of the problem depends on the information available to each agent. In this paper, we consider two settings. In the simpler setting, we assume that each agent is aware of its own reward profile in advance. We refer to this as the offline BLOCKMATCH problem. Additionally, we consider and online version of BLOCKMATCH with bandit feedback. In the bandit version of BLOCKMATCH, we assume that agents are not aware of their reward profiles in advance, but after each time step observes the service they were matched to and the reward it received from being matched. As a result, agents must tradeoff learning their own reward profile through the submission of suboptimal preference orderings, with exploiting their current knowledge regarding their true reward profile.

3 The Offline BLOCKMATCH Problem

We start with the analysis of the offline BLOCKMATCH problem. We first provide a lower on the distortion achieved by both randomised and deterministic policies. Then, we briefly discuss why trivial extensions of truthful one-shot one-sided matching algorithms do not result in truthful mechanisms[1]. Instead, we focus on designing mechanisms which use truthful one-shot one-sided matching mechanisms as basis, and have bounded incentive ratio (see below), a notion weaker than truthfulness. More precisely, we describe the `RRSD` algorithm. We show that the incentive ratio of `RRSD` is bounded by $1 - 1/e$, and provide upper bounds on the distortion of `RRSD`, which match our lower bounds.

3.1 Lower Bounds on the Distortion of Randomised and Deterministic Policies

To evaluate the efficacy of a given policy we use distortion, a standard notion of approximation for settings with ordinal preferences.

Definition 2. *The distortion of a policy is the worst-case ratio (over all possible instances of the offline BLOCKMATCH problem) between the expected social welfare of the matching sequence, π, returned by the policy, and the social welfare of the optimal matching sequence, $M_T^*(\mu, D) \in \mathcal{M}_T^D$, under the assumption that agents will submit ordinal preferences truthfully:*

$$\sup_{\mu,\, D} \frac{SW(M_T^*(\mu, D))}{\mathbb{E}\left[SW(\pi)\right]}$$

Note that distortion is the approximation ratio of the policy π with respect to best matching sequence.

First, we prove that the distortion of any deterministic policy is $\Omega(S)$. That is, the distortion of any deterministic policy scales linearly with the number of services in the best case. In the proof, we first carefully construct a set of ordinal preferences. Then, given any matching sequence M_T, we show that there exists a set of reward profiles which induces the aforementioned ordinal preferences and on which M_T incurs distortion of order $\Omega(S)$. We defer the full proof to the appendix.

[1] By truthful we simply mean that, in order to maximise their own utilities, agents are motivated to submit a linear preference ordering induced by their reward profile.

Theorem 1. *The distortion of any deterministic policy is* $\Omega(S)$.

Next, we prove that the distortion incurred by any randomised policy is $\Omega(\sqrt{S})$. To prove this, we first show that it is sufficient to consider only *anonymous* policies. Then, we construct a set of reward profiles which yields the desired distortion for all anonymous truthful policies. Once again, we defer the full proof to the appendix.

Theorem 2. *The distortion of the best randomised policy is* $\Omega(\sqrt{S})$.

3.2 Constructing Truthful Algorithms for the Offline BLOCKMATCH Problem

In the offline setting, each agent is aware of her expected rewards and the ordinal preference. However, agents may still choose to submit false preferences, with the aim of increasing their own utilities. As a result, the distortion incurred by a given policy may not reflect its performance in practice. Note that in standard one-shot one-sided matching problems, this issue is sidestepped via the employment of truthful mechanisms. That is, mechanisms that are constructed so that it is in the best interest of each agent to submit their preferences truthfully, like RSD. In addition, the restriction to considering truthful algorithms is well justified by the revelation principle. In a similar vein, we would like to develop truthful algorithms for the offline BLOCKMATCH setting.

One may be tempted to apply such truthful one-shot mechanisms to our setting directly. That is, to apply an algorithm such as RSD repeatedly at the start of each time step. This intuition is correct when there is no blocking, as the matching problems for time steps are then independent of each other. However, with blocking, the matchings from previous rounds will have a substantial effect on the set of valid matchings in future rounds. As a result, immediately obvious approaches, such as matching according to RSD repeatedly, do not result in truthful algorithms (see the appendix for an example). One simple way of generating truthful policies is to run a truthful one-shot one-sided matching mechanism once every $\tilde{D}$ time steps and simply return the empty matching $\emptyset$ in the remaining time steps. Such an approach decouples each time step from the next, resulting in truthfulness, but comes at the cost of only matching in at most $\lceil T/\tilde{D} \rceil$ rounds.

Instead, we construct an algorithm for the offline BLOCKMATCH problem from truthful one-shot matching algorithms in a different manner. More specifically, we propose the Repeated Random Serial Dictatorship (`RRSD`) algorithm, which uses RSD as a basis. Whilst `RRSD` is not truthful, it does have a bounded incentive ratio, a form of approximate truthfulness which will be defined shortly.

We write $\mu_i \rhd o_i$ to denote a reward profile μ_i consistent with the ordering o_i i.e. $a \succ_o b$ implies $\mu_{i,a} \geq \mu_{i,b}$. Let o_i^* denote the linear preference ordering induced by the reward profile μ_i. In other words, o_i^* represents agent i's true preferences. We write $\pi(O_i^*, O_{-i}^*)$ to denote the (potentially random) matching sequence returned by policy π when all agents report truthfully in each round. Similarly, let $\pi(O_i, O_{-i}^*)$ denote the (again, potentially random) matching sequence returned by policy π when agent i submits preferences $O_i = (o_{i,1}, \ldots, o_{i,T})$ and all remaining agents report their preferences truthfully in each round. We define the incentive ratio of a given policy π as the worst-case ratio between the expected utility an agent receives when it reports truthfully, and the maximum expected utility attainable.

Definition 3. *The incentive ratio* $\zeta(\pi) \in \mathbb{R}_+$ *of a matching policy* π *is given by:*

$$\zeta(\pi) = \max_{D,\, O_{-i}^*,\, \mu_i \rhd o_i^*} \frac{\mathbb{E}[\, W_i(\pi(O_i^*, O_{-i}^*))]}{\max_{O_i} \mathbb{E}[\, W_i(\pi(O_i, O_{-i}^*))]} \tag{1}$$

The policy's incentive ratio tells you how much an agent can gain via lying about their preferences.

Algorithm 1: RRSD

Input : T, $\mathcal{K}$, D, $\mathcal{S}$
Output: M_T // A greedy solution to the BLOCKMATCH Problem

$M_T = (m_t)_{t=1}^T = (\emptyset)_{t=1}^T$ // Start with the empty matching sequence
Set $\mathcal{C} = \mathcal{K}$ // Initialise set of agents which have not yet been matched
Observe $(o_{1,1}, \ldots, o_{N,1})$
while $\mathcal{C} \neq \emptyset$ **do**
    Sample i from $\mathcal{C}$ uniformly at random
    for $t = 1, \ldots T$ **do**
        /* Assign the best available service at time step t to agent i according to her own preferences */
        Set $m_t(i) = A(o_{i,1}, M_T, t)$;
    end
    $\mathcal{C} = \mathcal{C} \setminus \{i\}$
end
return M_T

3.3 A Greedy Algorithm for the Offline BLOCKMATCH Problem

We are now ready to define the RRSD algorithm. In the first time step, RRSD observes the ordinal preferences $(o_{1,1}, \ldots, o_{N,1})$submitted by each agent. In what follows, RRSD will slowly build up a matching sequence $M_T = (m_t)_{t=1}^T$ over time by iterating through agents and time steps. In other words, RRSD begins with the empty matching sequence, where $m_t = \emptyset$ for all t.

To begin building up the matching sequence M_T, RRSD samples an agent i uniformly at random without replacement. RRSD then iterates through each time step in order. At time step t, RRSD updates M_T by assigning the agent service $A(o_{i,1}, M_T, t)$, where $A(o_{i,1}, M_T, t)$ denotes the best available service according to the preference ordering submitted by agent i which is not blocked at time step t in the current version of M_T. RRSD repeats this process until no agents remain and returns the finished matching sequence M_T. The pseudocode for RRSD is given in Algorithm 1. In summary, RRSD greedily assigns services to each agent for the entire time horizon in a random order.

Next, we show that RRSD has bounded incentive ratio. In particular the incentive ratio is bounded by $1 - 1/e$ as $T \to \infty$.

Theorem 3. *The incentive ratio of RRSD is asymptotically bounded below by* $1 - 1/e$.

The main idea behind the proof is as follows. Assume that once agent i is chosen by the RRSD, it is free to choose any sequence of services to match to, as long as the previous blocking constraints imposed by the current state of the matching sequence are not violated. Observe that this is simply an offline blocking bandits problem with additional blocking constraints imposed on the agent by the current state of the matching sequence. The sequence of services proposed by RRSD represents a greedy solution to this problem. We then prove that such a greedy solution achieves an approximation ratio of $1 - 1/e$ asymptotically, which directly yields the result of the theorem. We defer the full proof to the appendix.

Furthermore, under the assumption that agents submit their preferences truthfully, we prove an upper bound on the distortion of RRSD, that matches the lower bound on the distortion of any truthful policy as described in Theorem 2.

Theorem 4. *The distortion of* `RRSD` *is at most* $O(\sqrt{S})$.

We now show that it is possible to match the lower bound with a deterministic policy. In fact, the derandomized version of `RRSD` has distortion of at most $\mathcal{O}(S)$. The main idea is that we can select $O(N^2 \log N)$ permutations so that the number of times agent i is selected at j-th position is $\Theta(N \log N)$. Then we can run through these permutations one by one instead of selecting one permutation uniformly at random. The details of the algorithm are provided in the appendix.

Theorem 5. *There is a deterministic policy with distortion at most* $O(S)$ *for any* $T \geq O(N^2 \log N)$.

Note that, in practice, `RRSD` is not guaranteed to have an expected social welfare which matches its own distortion upper bound, as `RRSD` is not truthful. However, if agents are lazy, in the sense that they only care about receiving at least a $1 - 1/e$ fraction of their optimal utility under `RRSD`, we can assume that each agent will report truthfully. In such cases, `RRSD` will have an expected social welfare matching its own distortion upper bound.

4 BLOCKMATCH with Bandit Feedback

We now turn our attention to the online BLOCKMATCH setting with bandit feedback. In this setting, we assume that each agent is initially unaware of it true preference ordering. At the start of each time step t, each agent $i \in \mathcal{K}$ simultaneously submits a preference ordering over services to a central mechanism. The central mechanism then, according to a potentially randomised policy $\pi \in \mathcal{P}$, allocates each agent i a unique service $\pi(i, t) \in \mathcal{S}$.

In the bandit setting, we have an additional layer of difficulty caused by the learning part (i.e., due to the uncertainty about the true ordering at the beginning, the algorithm might make several mistakes until it gets a good estimate of the true ordering). A performance metric that is typically used in the bandit literature to measure the efficiency of such learning algorithms in such uncertain settings is the regret, which is defined as a difference between the performance of a learning algorithm and that of the best fixed policy in hindsight (i.e., the policy that allocates the same service to the same agent when it's available). As we have already shown, even in the offline setting, finding the optimal policy π^* is computationally intractable. Thus, we cannot expect this best fixed in hindsight policy to have good performance in general (compared to that of the offline optimal), and therefore, would not serve as a good benchmark. As a result, we shall use the following regret definition instead:

Dynamic approximate regret. We compare the performance of a policy to the dynamic oracle algorithm which returns the optimal offline solution of BLOCKMATCH. We define the α-regret under a policy $\pi \in \mathcal{P}$ as the difference between an (offline) α-optimal policy and the expected performance under policy π. More precisely, let π^* denote the policy of the dynamic oracle algorithm. The α-regret of a policy $\pi \in \mathcal{P}$ against π^* is defined to be

$$R_\pi^\alpha(D, \mu, T) = \alpha \mathbb{E}[\mathrm{SW}(\pi^*)] - \mathbb{E}[\mathrm{SW}(\pi)]$$

where the expectation is taken over all possible randomisation of the policy π^*.This definition is useful if we want to use an offline algorithm with provable distortion guarantees. For example, if we compare the performance of our learning algorithm against an offline mechanism with α distortion, then the difference is in fact a dynamic $1/\alpha$-regret.

We are also interested in the performance of each single agent in this setting. Overall, if an allocation cannot guarantee personal performance fitness for each agent, then it is unlikely that the agents will follow that mechanism. To this end, we define the following:

Approximate incentive compatible regret. We recall some of the notations from Section 3. In particular, we denote by O_i the collection of reports agent i sends to the mechanism during its whole scope. Similarly, O_{-i} is the collection of reports agents other than i sends to the mechanism. Finally, we denote by $\pi(O_i, O_{-i})$ the allocation done by policy π and caused by the received reports O_i and O_{-i}, respectively. For a given π, we define agent i's α-IC regret (or α incentive compatible regret) for reporting report O_i as:

$$R^{\alpha}_{\pi,z}(D, \mu, T) = \alpha \max_{O'_i} \mathbb{E}[r(\pi(O'_i, O_{-i}))] - \mathbb{E}[r(\pi(O_i, O_{-i}))]$$

Ideally, we aim to design a mechanism where truthful reporting enjoys a bounded α-IC regret.

We now turn to the design of learning algorithms for the SBM problem in the bandit setting. The key challenge in this setting is that the agents need to both estimate their own true preference ordering as well as to optimise their total rewards and utilities over the finite time horizon T. To do so, agents can turn to bandit algorithms, which typically fall within two classes, namely: (i) explore-then-commit (ETC) [31, 20], where agents dedicate the first block of time steps to purely learn the ordering (i.e., to explore), and then use that estimated ordering to perform requests for the allocations (the commit phase); and (ii) sequential approach [8, 1], where agents both sequentially update their estimates and optimize without clear separated blocks of these two tasks. It is easy to show that if we apply the latter to our setting, it would be very difficult to guarantee (approximate)-truthfulness. In particular, agents could strategically lie about their current estimates in order to influence the future allocations that may lead to better exploration or exploitation. As we conjecture it is not possible to derive meaningful incentive ratio and other approximate-truthfulness guarantees for this case, we opt for the ETC based approach in this paper.

Our central mechanism can be described as follows. We consider two phases, in the first phase, the mechanism lets the agents learn their ordering. We refer to this as the exploration phase. At the beginning of the second phase, the central mechanism will ask the agents to submit their ordering, learnt during the exploration phase. The mechanism then runs `RRSD` with the reported orderings.

While the second phase is straightforward given the description of `RRSD` in Section 3, it is not trivial how to efficiently coordinate the allocations between the agents in the first phase. In particular, the first challenge is to decide when to stop the exploration phase. The second challenge comes from the blockings. That is, how to allocate the services to the agent such that the total time spent on the exploration phase is minimised. This is crucial shorter the exploration phase is, the smaller the regret will be [20, 31]. To tackle both challenges, we propose the following protocol.

Multi-agent learning-to-rank. Suppose each agent i has their favourite learning-to-rank algorithm [30, 29, 36], denoted as BASE_i, which is used to learn the service ordering of agent i. We assume that each BASE_i possesses the following properties (which are common in most of the learning-to-rank algorithms in the literature):

- It runs in epochs. Within each epoch, the number of pulls per each arm is specified at the beginning of that epoch (epoch length can also be 1, in that case it only pulls 1 single arm).
- There is a stopping condition which terminates the algorithm, and the algorithm terminates almost surely. Let τ_i denote the random variable that captures the total number of samples (i.e., the total sum of allocation of agent i over all the services) until termination.
- There is an algorithm specific coefficient $H_i > 0$ such that the following holds: For any $\delta \in (0, 1)$ we have $P(\tau_i \leq H_i \log(1/\delta)) > 1 - \delta$.

Given these assumptions, we now turn to the description of the multi-agent learning-to-rank protocol, which consists of the following steps: We run in epochs. For each epoch r we maintain a set of active agents who are still participating in that epoch. For $r = 1$, all the agents are active. Then for each current epoch $r \geq 1$:

1. Let $n_{i,j}(r)$ denote the total number of times active agent i requests to be allocated to service j within this epoch (this is determined by BASE_i).

2. We fix the order of the agents in some arbitrary way. We also fix the order of the services per agent, also in an arbitrary way.

3. We then use the following simple greedy approach to schedule the set of $\{n_{i,j}(r)\}_i$ allocation requests. At each time step t within the current epoch r, we go through the agents in the order above and we assign the first available service (i.e., no blocking and no collision with the others) to that agent if there are any.

4. We stop the epoch when all the allocation requests from $n_{i,j}(r)$ have been allocated and all the services are available again for all the agents (this would allow us to start the next epoch without any blockings).

5. If BASE_i indicates that agent i should stop then agent i reports this to the central mechanism, which then eliminates agent i from the *list of active agents*. The whole protocol stops when there are no more active agents. Otherwise, it moves to the next epoch with the remaining active agents.

We refer to this protocol as synchronous multi-agent learning-to-rank (sync-MALTR) as opposed to its asynchronous version. In particular, asynchronous multi-agent learning-to-rank (async-MALTR) does not need to wait until all the services are available again to move to the next epoch, but allows each individual agent to move to their own next epoch as soon as their allocation requests in the previous epoch have been fulfilled.

Regret analysis. We now turn to the performance analysis of our central mechanism, that uses multi-agent learning-to-rank protocol for exploration, and the RRSD for the second phase.

Theorem 6. *The following statements are true:*
(i) The expected dynamic $1/\sqrt{S}$*-regret of the central mechanism is at most* $O(\tilde{D}\sum_i H_i \log(NT))$.
(ii) For each agent i*, reporting truthfully after the exploration phase would yield an expected* $(1-1/e)$*-IC regret of* $O(H_i \log(NT))$.
(iii) Using async-MATLR, the number of time steps needed in the exploration phase is at most twice as many as the optimal allocation requires.

Note that since the BASE algorithms are quite generic, it is difficult to improve the $\tilde{D}\sum_i H_i$ coefficient in the regret bound. However, we will show in the appendix that by using more specific learning-to-rank algorithms, this value can be significantly improved.

5 Conclusions and Future Work

In this paper, we introduced the sequential blocking matching (SBM) model to capture repeated one-sided matching with blocking constraints. Our offline benchmarks chart the performance of

deterministic and randomized policies in terms of distortion and incentive ratio. On the other hand, our online algorithm provides sublinear dynamic approximate regret.

There are many interesting directions for future work. It would be interesting to consider a two-sided matching problem [42] where the services / items also have preferences over the agents and such preferences need to be learned over time. Another important direction is the design of decentralized mechanisms where the learning agents choose services independently. Finally, we assumed that the preferences are static over time, but the agents may not be aware of them. In practice, the agents' preferences can change with time [11], and such dynamics need to be incorporated in the design of repeated matching algorithms.

References

[1] Shipra Agrawal and Navin Goyal. "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on learning theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 39–1 (cit. on p. 10).

[2] Eman Aldhahri, Vivek Shandilya, and Sajjan Shiva. "Towards an Effective Crowdsourcing Recommendation System: A Survey of the State-of-the-Art". In: *2015 IEEE Symposium on Service-Oriented System Engineering*. 2015, pp. 372–377 (cit. on p. 2).

[3] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A Voudouris. "Peeking behind the ordinal curtain: Improving distortion via cardinal queries". In: *Artificial Intelligence* 296 (2021), p. 103488 (cit. on p. 4).

[4] Asim Ansari, Skander Essegaier, and Rajeev Kohli. "Internet Recommendation Systems". In: *Journal of Marketing Research* 37.3 (2021/05/23 2000), pp. 363–375 (cit. on p. 2).

[5] Elliot Anshelevich, Onkar Bhardwaj, Edith Elkind, John Postl, and Piotr Skowron. "Approximating optimal social choice under metric preferences". In: *Artificial Intelligence* 264 (2018), pp. 27–51 (cit. on p. 4).

[6] Elliot Anshelevich and John Postl. "Randomized social choice functions under metric preferences". In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 797–827 (cit. on p. 4).

[7] Elliot Anshelevich and Shreyas Sekar. "Blind, greedy, and random: Algorithms for matching and clustering using only ordinal information". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016 (cit. on p. 4).

[8] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2-3 (2002), pp. 235–256 (cit. on p. 10).

[9] Soumya Basu, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. "Contextual Blocking Bandits". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. 2021, pp. 271–279 (cit. on p. 3).

[10] Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. "Blocking Bandits". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. 2019 (cit. on pp. 3, 18, 20, 24).

[11] Dirk Bergemann and Juuso Välimäki. "Dynamic mechanism design: An introduction". In: *Journal of Economic Literature* 57.2 (2019), pp. 235–74 (cit. on p. 12).

[12] Nicholas Bishop, Hau Chan, Debmalya Mandal, and Long Tran-Thanh. "Adversarial Blocking Bandits". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. 2020, pp. 8139–8149 (cit. on p. 3).

[13] Ilai Bistritz, Tavor Baharav, Amir Leshem, and Nicholas Bambos. "My fair bandit: Distributed learning of max-min fairness with multi-player bandits". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 930–940 (cit. on p. 3).

[14] Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D Procaccia, and Or Sheffet. "Optimal social choice functions: A utilitarian view". In: *Artificial Intelligence* 227 (2015), pp. 190–213 (cit. on p. 4).

[15] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. "The complexity of Unique k-SAT: An Isolation Lemma for k-CNFs". In: *Journal of Computer and System Sciences* 74.3 (2008). Computational Complexity 2003, pp. 386–393. ISSN: 0022-0000 (cit. on pp. 24, 25).

[16] Ning Chen, Xiaotie Deng, Hongyang Zhang, and Jie Zhang. "Incentive Ratios of Fisher Markets". In: *Automata, Languages, and Programming*. Ed. by Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 464–475. ISBN: 978-3-642-31585-5 (cit. on p. 3).

[17] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. "Pick-a-Crowd: Tell Me What You like, and i'll Tell You What to Do". In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pp. 367–374 (cit. on p. 2).

[18] Steven N. Durlauf and Lawrence Blume. *The new Palgrave dictionary of economics*. Palgrave Macmillan, 2008 (cit. on p. 2).

[19] Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, and Jie Zhang. "Social welfare in one-sided matchings: Random priority and beyond". In: *International Symposium on Algorithmic Game Theory*. Springer. 2014, pp. 1–12 (cit. on pp. 4, 17, 22).

[20] Aurélien Garivier, Tor Lattimore, and Emilie Kaufmann. "On explore-then-commit strategies". In: *Advances in Neural Information Processing Systems* 29 (2016), pp. 784–792 (cit. on p. 10).

[21] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. "The pinwheel: a real-time scheduling problem". In: *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track*. Vol. 2. 1989, 693–702 vol.2 (cit. on pp. 24, 25).

[22] Hadi Hosseini, Kate Larson, and Robin Cohen. "Matching with Dynamic Ordinal Preferences". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (2015) (cit. on p. 4).

[23] Hadi Hosseini, Kate Larson, and Robin Cohen. "On Manipulablity of Random Serial Dictatorship in Sequential Matching with Dynamic Preferences". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (2015) (cit. on p. 4).

[24] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. "Recommendation systems: Principles, methods and evaluation". In: *Egyptian Informatics Journal* 16.3 (2015), pp. 261–273 (cit. on p. 2).

[25] Tobias Jacobs and Salvatore Longo. *A New Perspective on the Windows Scheduling Problem*. 2014. arXiv: 1410.7237 (cit. on p. 25).

[26] B. Kalyanasundaram and K. Pruhs. "Online Weighted Matching". In: *Journal of Algorithms* 14.3 (1993), pp. 478–488 (cit. on p. 4).

[27] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. "Online Bipartite Matching with Unknown Distributions". In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. 2011, pp. 587–596 (cit. on p. 4).

[28] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. "An Optimal Algorithm for On-Line Bipartite Matching". In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. 1990, pp. 352–358 (cit. on p. 4).

[29] Nikolai Karpov and Qin Zhang. "Batched Coarse Ranking in Multi-Armed Bandits". In: *Advances in Neural Information Processing Systems* 33 (2020) (cit. on p. 10).

[30] Sumeet Katariya, Lalit Jain, Nandana Sengupta, James Evans, and Robert Nowak. "Adaptive sampling for coarse ranking". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1839–1848 (cit. on p. 10).

[31] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. "On the complexity of best-arm identification in multi-armed bandit models". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1–42 (cit. on p. 10).

[32] David Kempe. "Communication, distortion, and randomness in metric voting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 2087–2094 (cit. on p. 4).

[33] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. "On-line algorithms for weighted bipartite matching and stable marriages". In: *Theoretical Computer Science* 127.2 (1994), pp. 255–267 (cit. on p. 4).

[34] Robert Kleinberg and Nicole Immorlica. "Recharging bandits". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 309–319 (cit. on p. 3).

[35] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. "Regret bounds for sleeping experts and bandits". In: *Machine learning* 80.2 (2010), pp. 245–272 (cit. on p. 3).

[36] Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvári. "TopRank: A practical algorithm for online stochastic ranking". In: *NeurIPS*. 2018 (cit. on p. 10).

[37] Lydia T Liu, Horia Mania, and Michael Jordan. "Competing bandits in matching markets". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 1618–1628 (cit. on p. 3).

[38] Debmalya Mandal, Ariel D Procaccia, Nisarg Shah, and David P Woodruff. "Efficient and thrifty voting by any means necessary". In: *Advances in Neural Information Processing Systems* (2019) (cit. on p. 4).

[39] Gergely Neu and Michal Valko. "Online combinatorial optimization with stochastic decision sets and adversarial losses". In: *Neural Information Processing Systems*. 2014 (cit. on p. 3).

[40] David C. Parkes and Ariel D. Procaccia. "Dynamic Social Choice with Evolving Preferences". In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. 2013, pp. 767–773 (cit. on p. 4).

[41] Ariel D Procaccia and Jeffrey S Rosenschein. "The distortion of cardinal preferences in voting". In: *International Workshop on Cooperative Information Agents*. Springer. 2006, pp. 317–331 (cit. on pp. 3, 4).

[42] Alvin E Roth and Marilda Sotomayor. "Two-sided matching". In: *Handbook of game theory with economic applications* 1 (1992), pp. 485–541 (cit. on p. 12).

[43] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. "Kidney Exchange*". In: *The Quarterly Journal of Economics* 119.2 (May 2004), pp. 457–488 (cit. on p. 2).

[44] Abishek Sankararaman, Soumya Basu, and Karthik Abinav Sankararaman. "Dominate or Delete: Decentralized Competing Bandits in Serial Dictatorship". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1252–1260 (cit. on p. 3).

[45] Benjamin Satzger, Markus Endres, and Werner Kießling. "A Preference-Based Recommender System". In: *E-Commerce and Web Technologies*. Ed. by Kurt Bauknecht, Birgit Pröll, and Hannes Werthner. 2006, pp. 31–40 (cit. on p. 2).

[46] Gerhard J Woeginger. "The Open Shop Scheduling Problem". In: *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018 (cit. on p. 24).

A Proofs for Section 3

A.1 Proof of Theorem 1

Proof. Without loss of generality we can assume that $N = S$. In case $N < S$, our lower bound example can be extended by assuming that all the players have value 0 for $S - N$ services.

Therefore, we now consider an instance of BLOCKMATCH with N agents and N services, where each agent has the same preferences. For simplicity assume that service 1 is most preferred, service 2 the next, and so on, until service N. Furthermore, assume that the blocking delay on each of the N services is $\tilde{D}$. We add additional services as needed to ensure that a service is always available to each agent. Note that we need to add at most N services with no blocking delays. All of these 'empty' services appear at the end of the agents' preferences. Given a matching sequence, π, output by a policy, we will assign rewards vectors, μ_i, to each agent i, which induce the preference ordering, in the following manner.

There must exist some agent, $i_1 \in \mathcal{K}$, who is assigned service 1 at most $T/\tilde{D}N$ times. We set $\mu_{i_1} = (1, 0, \ldots, 0)$. Discarding agent i_1, there must exist an agent, i_2, who is assigned service 1 or 2 at most $T/\tilde{D}(n-1)$ times. We set $\mu_{i_2} = (1/2, 1/2, 0, \ldots, 0)$. Again, we discard agent i_2, and find the agent i_3 who has been assigned services 1, 2 or 3 at most $T/\tilde{D}(N-2)$ times, and assign the reward vector $\mu_{i_3} = (1/3, 1/3, 1/3, 0 \ldots, 0)$. We proceed in this manner for a total of N steps until all agents are assigned preferences.

Given the assigned reward vectors, it is obvious that an optimal matching sequence assigns service j to agent i_j whenever the service is available. This optimal matching sequence achieves a total expected welfare of order $O(\log(N)T/\tilde{D})$. In comparison, the matching proposed by the policy achieves welfare at most $O(\log(N)T/\tilde{D}N)$. As a result, for any matching sequence returned by the policy, there is a set of reward vectors such that the distortion is of order $O(N) = O(S)$.

□

A.2 Proof of Theorem 2

Proof. By an argument similar to the proof of Theorem 1, without loss of generality we can assume there are N agents and N services. We will construct an example where all the blocking lengths D_{ij}-s are identical (e.g. say D). First, we show that it is sufficient to consider *anonymous* mechanisms. Given a preference profile o, let $A_{ij}(o) \in \{0, 1, \ldots, T\}$ be a random variable that indicates the number of times agent i was allocated service j. We will call a randomised matching algorithm anonymous if $\mathbb{E}[A_{ij}(o_1, \ldots, o_N)] = \mathbb{E}[A_{\sigma(i)j}(o_{\sigma(1)}, \ldots, o_{\sigma(N)})]$ for all permutations σ. Suppose we are given a matching algorithm that has distortion at most ρ i.e. $\sum_{ij} \mu_{ij}\mathbb{E}[A_{ij}(o)] \geq \rho\mathrm{OPT}(\mu)$. Now we consider a new matching algorithm that selects a permutation σ uniformly at random and then applies the same algorithm on the input $o_\sigma = (o_{\sigma(1)}, \ldots, o_{\sigma(N)})$. Then the expected social welfare of the new algorithm is

$$\mathbb{E}_\sigma \left[\sum_{ij} \mu_{\sigma(i)j} A_{\sigma(i)j}(o_\sigma) \right] \geq \mathbb{E}_\sigma \left[\rho\mathrm{OPT}(\mu_\sigma) \right] = \rho\mathrm{OPT}(\mu)$$

The first inequality follows because the original mechanism gives ρ distortion even when applied to the profile μ_σ and the second equality follows because the optimal welfare ($\mathrm{OPT}(\mu) = \sum_{ij} \mu_{ij} A^*_{ij}$) is invariant to permutation. Therefore, the new anonymous mechanism has distortion at most ρ.

We now construct a reward profile which is very similar to the one constructed in the proof of Lemma 8 of [19]. Consider the following reward profile.

$$\forall i \in [\sqrt{N}],\ \mu_{i,j} = \begin{cases} 1 - \sum_{j \neq i} \mu_{i,j} & \text{if } j = i \\ \frac{N-j}{10N^3D} & \text{o.w.} \end{cases}$$

$$\forall \ell \in [\sqrt{N}-1],\ \mu_{i+\ell\sqrt{N},j} = \begin{cases} 1 - \sum_{j \neq i} \mu_{i,j} & \text{if } j = i \\ \frac{1}{\sqrt{N}} - \frac{j}{10N^2} & \text{if } j \neq i \ \ \& \ \ j \leq \sqrt{N} \\ \frac{n-j}{10N^3D} & \text{o.w.} \end{cases}$$

The N agents are grouped into $\sqrt{N}$ groups and all agents in group i have the same preference order. Let $G_i = \{i\} \cup \left\{i + \ell\sqrt{N} : \ell = 1, \ldots, \sqrt{N}-1\right\}$. Then all the agents in group G_i have preference order $i \succ 1 \succ \ldots \succ i-1 \succ i+1 \succ \ldots \succ N$. Therefore, for any item j, all the agents in group G_i have the same expected number of allocations. Let us call this number of allocations T_{ij}. Since any item j can be allocated at most T/D times we have

$$\sum_{i=1}^{\sqrt{N}} \sum_{p \in G_i} T_{ij} \leq \frac{T}{D} \ \Rightarrow \sum_{i=1}^{\sqrt{N}} T_{ij} \leq \frac{T}{D\sqrt{N}} \tag{2}$$

We now bound the expected welfare of any randomized and anonymous matching algorithm with the given reward profile. For any agent $i \in [\sqrt{N}]$, the maximum expected utility over the T rounds is at most $T_{ii} + \sum_{j \neq i} T_{ij} \frac{N-j}{10n^3D} \leq T_{ii} + O\left(\frac{T}{ND}\right)$. Now consider an agent $i + \ell\sqrt{N}$ for $\ell \in [\sqrt{N}-1]$. Such an agent's utility over the T rounds is at most $T_{ii} O\left(\frac{1}{\sqrt{N}}\right) + \sum_{j \neq i, j \leq \sqrt{N}} T_{ij} \frac{1}{\sqrt{N}} + \sum_{j > \sqrt{N}} T_{ij} \frac{N-j}{10N^3D} \leq O\left(\frac{1}{\sqrt{N}}\right) \sum_{j=1}^{\sqrt{N}} T_{ij} + O\left(\frac{T}{ND}\right)$. Therefore, the total utility over all the N agents is bounded by

$$\begin{aligned} &\sum_{i=1}^{\sqrt{N}} T_{ii} + O\left(\frac{1}{\sqrt{N}}\right) \sum_{i=1}^{\sqrt{N}} \sum_{\ell=1}^{\sqrt{N}-1} \sum_{j=1}^{\sqrt{N}} T_{ij} + O\left(\frac{T}{D}\right) \\ &\leq \sum_{i=1}^{\sqrt{N}} T_{ii} + \sum_{i=1}^{\sqrt{N}} \sum_{j=1}^{\sqrt{N}} T_{ij} + O\left(\frac{T}{D}\right) \\ &\leq 2 \sum_{j=1}^{\sqrt{N}} \sum_{i=1}^{\sqrt{N}} T_{ij} + O\left(\frac{T}{D}\right) \\ &\leq 2 \sum_{j=1}^{\sqrt{N}} \frac{T}{D\sqrt{N}} + O\left(\frac{T}{D}\right) = O\left(\frac{T}{D}\right) \end{aligned}$$

The last line uses equation (2). On the other hand, any deterministic and non-anonymous allocation rule that always assigns item i agent i every D rounds achieves a welfare of at least $\sqrt{N}\frac{T}{D}(1 - \frac{1}{10ND}) \geq O\left(\frac{T\sqrt{N}}{D}\right)$. This establishes a bound of $O(\sqrt{N}) = O(\sqrt{S})$ on distortion. □

A.3 Proof of Theorem 3

Proof. Without loss of generality, assume that agent k is selected at random in the kth position by RRSD. Assume, for the moment, that agents 1 to $k-1$ are not allocated any services. Additionally, suppose that agent k is free to choose its own allocation of services independent of the RRSD algorithm. Under these assumptions, agent k is posed with an offline blocking bandits problem as described in [10]. The solution proposed by RRSD corresponds to a greedy approach in which the best service available is allocated at each time step. Thus, proving that such a greedy algorithm has an approximation ratio of $1 - \frac{1}{e}$ implies the result in this restricted case. This fact was proven in [10].

We will show that this result holds more generally, regardless of the allocations chosen by RRSD in previous time steps. Again, assume agent k is free to choose its allocation, independent of RRSD. That is, agent k is tasked with solving the following integer linear programming problem (ILP):

$$\begin{aligned}
\max_{x_{t,s}} \quad & \sum_{t=1}^{T} \sum_{s=1}^{n} \mu_{k,s} x_{t,s} \\
\text{s.t.} \quad & x_{t,s} \in \{0,1\} \\
& y_{t,s} + x_{t,s} \leq 1 \quad \forall t \in [T], \forall s \in \mathcal{S} \\
& \sum_{s=1}^{N} x_{t,s} = 1 \quad \forall t \in [T] \\
& \sum_{t \in [D_{k,s}]} x_{t+t_0,s} \leq 1 \quad \forall t_0 \in T, \forall s \in \mathcal{S}
\end{aligned}$$

The variables $x_{t,s}$ indicate whether agent k is matched to service s at time step t. Meanwhile, the constants $y_{t,s}$ indicate whether agent k cannot be matched to service s due to delay constraints imposed by allocations of service s to agents 1 through $k-1$. The second set of constraints ensure that the matches chosen by agent k do not breach the delay constraints imposed by preexisting matches between services and agents 1 to $k-1$. The third set of constraints ensure that agent k may only be matched to one service at each time step. Lastly, the fourth set of constraints ensure that agent k chooses a sequence of matches which obeys its own delay constraints.

Computing the upper bound of the optimal solution. We derive an upper bound for this ILP through a series of relaxations. First of all, we relax the integer constraints, so that at each time step agent k can match to a fractional mixture of services. Additionally, we replace the constants $y_{t,s}$ with variables $z_{t,s}$ constrained to lie in $[0,1]$. The idea in introducing these variables is to remove the blocking constraints imposed by the previous players and replace it with a constraint that stipulates that the total reduction in the time horizon available for agent k to match itself each service s must remain the same. That is, agent k is free to fractionally redistribute the blocked parts of the time

horizon imposed by the previous $k-1$ agents. This results in the following linear program (LP):

$$\begin{aligned}
\max_{x_{t,s},\, z_{t,s}} \quad & \sum_{t=1}^{T}\sum_{s=1}^{N} \mu_{k,s} x_{t,s} \\
\text{s.t.} \quad & x_{t,s} \in [0,1] \\
& z_{t,s} + x_{t,s} \leq 1 \quad \forall t \in [T], \forall s \in \mathcal{S} \\
& \sum_{s=1}^{N} x_{t,s} = 1 \quad \forall t \in [T] \\
& \sum_{t=1}^{T} z_{t,s} = \sum_{t=1}^{T} y_{t,s} \quad \forall s \in \mathcal{S}, \forall j \in [k-1] \\
& \sum_{t \in [D_{k,s}]} x_{t+t_0,s} \leq 1 \quad \forall t_0 \in T, \forall s \in \mathcal{S}
\end{aligned}$$

It should be immediately obvious that this problem can be reformulated further, and, in fact, the individual fractional matches per time step can be replaced with fractional matches of agents to services for the entire time horizon. Similarly, the newly introduced auxiliary variables $z_{t,s}$ can be removed completely. In other words, it is clearly optimal for agent k to spread the blocked parts of the time horizon evenly across all time slots, and then greedily match services to itself in each time step whilst obeying its own delay constraints. Therefore, it only matters how often each service is matched to agent k, as the fractional amount matched for every time step will be the same. This leads us to the following, equivalent, LP reformulation:

$$\begin{aligned}
\max_{n_s} \quad & \sum_{s=1}^{N} n_s \mu_{k,s} \\
\text{s.t.} \quad & n_s \in [0, T/D_{k,s}] \\
& n_s + \sum_{t=1}^{T} y_{t,s} \leq T \quad \forall s \in \mathcal{S} \\
& \sum_{s=1}^{N} n_s = T
\end{aligned}$$

Additionally, we define $C_s = \{t \in [T] \,:\, y_{t,s} = 1\}$ as the set of time slots in which agent k cannot (in practice) be matched with service s because of delay constraints imposed by previous agents. Next, we show that this LP can be further formulated as a fractional bounded knapsack problem as follows:

Consider each s as an item with weight $D_{k,s}$ and value $\mu_{k,s}$, and n_s is the (fractional) number of times we pack item s into the knapsack (whose capacity is T). Note that the maximum value n_s can get in the previous LP is determined by the pattern of C_s, and is also capped by $T/D_{k,s}$. Therefore, in our bounded knapsack formulation, we can replace the constraints of n_s to be $n_s \leq T/D_{k,s} - b_s$ where b_s is the number of blocks caused by C_s. Note that in general $b_s \neq |C_s|$, as it heavily depends on the pattern of the blocks. Since $n_s \geq 0$, we have that $\frac{T}{D_{k,s}} \geq b_s$. Now, it is well known that this fractional bounded knapsack admits the optimal solution $\forall s \in \mathcal{S}, n_s^* = \min\{T/D_{k,s} - b_s, (T - b_s - \sum_{j=1}^{s-1} n_j^*)^+\}$.

Computing the lower bound of the greedy sequence of matches. Now consider the greedy sequence of matches for agent k generated by the `RRSD` algorithm. Let n_s^g denote the number

of times service s is matched to agent k by RRSD. Similarly let N_s denote the set of time slots in which agent k is allocated services 1 to $s-1$. The time slot where the periodic matching of service s to agent k collides with previous matches is denoted by $col_s = \{t \in N_s \cup C_s \ : \ D_{k,s} \mid t\}$ The number of times service s is matched to agent k is at least $\lceil (T - |col_s|)/D_{k,s} \rceil$. This holds because for service s we can remove the time slots with collisions and perform periodic placement perfectly with the remaining. $T - |col_s|$ time slots. Note that $|col_s| \leq \sum_{j=1}^{s-1} n_j^g + \sum_{t=1}^{T} y_{t,s} - |N_s \cap C_s|$.

We now define for each $s \in \mathcal{S}$, $n'_s = T_s/D_{k,s} - b_s$, where b_s is defined as the number of blocks caused by C_s from the previous agents, and $T_s = \left(T - \sum_{j=1}^{s-1} n'_j + |N_s \cap C_s|\right)^+$. That is, intuitively we can remove the time slots blocked by previous service matches and then match service s to agent k with period $D_{k,s}$. We claim that $\sum_{j=1}^{s} n_j^g \geq \sum_{j=1}^{s} n'_j$. In turn, this immediately implies that $\sum_{s=1}^{|\mathcal{S}|} n_s^g \mu_{k,s} \geq \sum_{s=1}^{|\mathcal{S}|} n'_s \mu_{k,s}$.

We prove the claim using induction on s. We know that $n_1^g \geq \lceil (T - b_1) D_{1,k} \rceil$, so the base case is satisfied. By the inductive hypothesis, assume that $\sum_{j=1}^{s} n_j^g \geq \sum_{j=1}^{s} n'_g$ for all $s < s'$. We have:

$$n_{s'}^g \geq \lceil (T - |col_{s'}|)/D_{k,s'} \rceil \geq \frac{1}{D_{k,s'}} \left(T - \sum_{j=1}^{s-1} n_j^g - b_{s'} + |N_{s'} \cap C_{s'}| \right)$$

$$= \frac{1}{D_{k,s'}} \left(T - \sum_{j=1}^{s-1} n'_j - \sum_{j=1}^{s-1} \left(n_j^g - n'_j \right) - b_{s'} + |N_{s'} \cap C_{s'}| \right) = n'_{s'} - \frac{1}{D_{k,s'}} \sum_{j=1}^{s-1} \left(n_j^g - n'_j \right)$$

Thus we have that $\sum_{j=1}^{s'} (n_j^g - n'_j) \geq (1 - 1/D_{k,s'}) \sum_{j=1}^{s'} (n_j^g - n'_j)$, which means $\sum_{j=1}^{s'} n_j^g \geq \sum_{j=1}^{s'} n'_j$, and thus the inductive hypothesis is proved.

Bounding the incentive ratio. Note that for any s, if $\frac{T}{D_{k,s}} = b_s$ then both the upper bound and lower bound solutions will not contain service s (as $n'_s \leq n^*_s = 0$). Therefore, w.l.o.g. we assume that $\frac{T}{D_{k,s}} > b_s$. We set $D'_{k,s}$ such that $\frac{1}{D'_{k,s}} = \frac{1}{D_{k,s}} - \frac{b_s}{T}$. With induction in s we can show that $n'_s = \frac{T}{D'_{k,j}} \prod_{j=1}^{s-1} (1 - \frac{1}{D'_{k,j}})$. In addition, we can also show that $n^*_s \leq \frac{T}{D'_{k,s}} + 1$. The lower bound for the incentive ratio then follows from the proof of [10] by optimsing over $D'_{k,s}$. This indicates that the incentive ratio is asymptotically bounded below by $1 - 1/e$. □

A.4 Description of DRRSD and Proof of Theorem 5

Algorithm 2 describes a deterministic sequential blocking matching algorithm. This algorithm is a derandomized version of Algorithm 1. By lemma 1 we can select a set of permutations so that each agent i becomes j th selected agent at least $1/2N$ times. Given such a collection of P permutations, algorithm 2 partitions T intervals into P groups, each of length T/P. At the start of group i, the algorithm orders the agents according to the i-th permutation, and allocates services in that order.

Lemma 1. *There exists a set of $4N^2 \log(N)$ permutations over N agents such that the fraction of times item i appears at the j-th position is at least $\frac{1}{2N}$.*

Proof. The proof is by the probabilistic method. Let us draw P permutation over the N items uniformly at random. Let X_{ij} be the fraction of times item i appears at j-th position over the P permutation. Then $\mathbb{E}[X_{ij}] = 1/N$. Moreover, from the Chernoff-Hoeffding inequality,

$$P\left(X_{ij} \leq \frac{1}{2N} \right) \leq 2e^{-2P\frac{1}{4N^2}} = 2e^{-\frac{P}{2N^2}}.$$

Algorithm 2: DRRSD (Derandomized RRSD)

Input : T, $\mathcal{K}$, D, $\mathcal{S}$, and a set of $P = 4N^2 \log(N)$ permutations $\{\sigma_1, \ldots, \sigma_P\}$
Output: M_T // A greedy solution to the BLOCKMATCH Problem
// Initialize matching sequence to the empty sequence
$M_T = (m_t)_{t=1}^T = (\emptyset)_{t=1}^T$
Observe $(o_{1,1}, \ldots, o_{N,1})$
for $t = 1, \ldots, T$ **do**
  if $t = \frac{(p-1)\times T}{P}$ **then**
    // Select p-th permutation σ_p.
    for $i = 1, \ldots, N$ **do**
      /* Agent $\sigma_p(i)$ is assigned her favourite service at time t among the available services. */
      Set $m_t(\sigma_p(i)) = A(o_{\sigma_p(i),1}, M_T, t)$
    end
    CUR $= p$
  end
  for $i = 1, \ldots, N$ **do**
    // Decide whether to unmatch agent i with service $m_t(i)$
    if $t + D_{i,m_t(i)} > \frac{CUR\times T}{P}$ **then**
      Set $m_t(i) = 0$
    end
  end
end
return M_T

Moreover, by a union bound over the N items and N positions we get that

$$P\left(\exists i, j \; X_{ij} \leq \frac{1}{2N}\right) \leq 2e^{-\frac{P}{2N^2}}.$$

Therefore, if $P \geq 4N^2 \log(N)$, the probability of observing a set of permutations such that each $X_{ij} \geq 1/2N$ is positive. This implies that if $P = 4N^2 \log(N)$, we can find a required set of permutations. □

A.5 Proof of Theorem 4

Proof. Our proof proceeds by upper bounding the distortion of RRSD by the distortion of RSD on a new reward profile. Let $\mathrm{SW}(\pi^\star, \mu)$ be the social welfare of the optimal policy $\pi^\star$ given the reward profile $\mu = (\mu_1, \ldots, \mu_n)$. Then the distortion of RRSD is given as

$$\rho = \sup_{\mu} \frac{\mathrm{SW}(\pi^\star, \mu)}{\mathbb{E}_{\pi \sim \mathtt{RRSD}}[\mathrm{SW}(\pi, \mu)]}$$

Consider the new utility profile $\tilde{\mu}_{i,j} = \mu_{i,j}/D_{i,j}$ for all $i \in [N]$ and $j \in [S]$. We will write $\mathrm{SW}_0(\sigma, \tilde{\mu})$ to denote the social welfare of a (one-shot) matching with the utility profile $\{\tilde{\mu}\}_{i=1}^N$. Then we claim that $\mathrm{SW}(\pi^\star, \mu) \leq T \cdot \mathrm{SW}_0(\sigma^\star, \tilde{\mu})$ for some (one-shot) matching $\sigma^\star$.

In order to see why the above claim is true, consider a different version of the online blocking matching problem where the arms are not blocked, and if we allocate arm j to agent i, we get a utility of $\tilde{\mu}_i(j)$. Given $\pi^\star$, we construct a new policy $\pi^\#$ for the new version of the blocked matching as follows. Whenever $\pi^\star_t(i) = j$, we set $\pi^\#_{t'} = j$ for $t' = t, t+1, \ldots, t + D_{i,j} - 1$. Note that this is always possible, as pulling arm j for agent i, makes that arm blocked for $D_{i,j}$ rounds in the original version of the problem. Moreover, if agent i gets utility of $u_i(j)$ from one allocation to arm j, the same agent gets utility of $D_{i,j} \times \tilde{\mu}_{i,j} = \mu_{i,j}$ under the new policy $\pi^\#$. This implies that $\mathrm{SW}(\pi^\star, \mu) = \mathrm{SW}(\pi^\#, \tilde{\mu})$. Now observer that under the new unblocked version of the problem there is no blocking constraint, so we can determine the best one-shot matching (say $\sigma^\star$) and apply it repeatedly over T rounds to get the optimal social welfare over the T rounds. This gives us $\mathrm{SW}(\pi^\star, \mu) = \mathrm{SW}(\pi^\#, \tilde{\mu}) \leq T \cdot \mathrm{SW}_0(\sigma^\star, \tilde{\mu})$.

We now prove $\mathbb{E}_{\pi \sim \texttt{RRSD}}[\mathrm{SW}(\pi, \mu)] = T\mathbb{E}_{\sigma \sim \texttt{RSD}}[\mathrm{SW}_0(\sigma, \tilde{\mu})]$. Suppose an agent i is assigned item j by the initial random draw of RRSD. Then agent i is assigned item j every $D_{i,j}$ rounds. Therefore, conditioned on the event, the total utility of agent i is $T/D_{i,j} \cdot \mu_{i,j}$. On the other hand, the event that agent i is assigned item j has the same probability under one-shot RSD, and agent i's utility in one round is $\tilde{\mu}_{i,j} = \mu_{i,j}/D_{i,j}$, which is exactly $1/T$ fraction of agent i's utility under RRSD, conditioned on the same event. As all the events of the form agent i is assigned item j have the same probability both under RRSD and RSD, we have $\mathbb{E}_{\pi \sim \texttt{RRSD}}[\mathrm{SW}(\pi, \mu)] = T\mathbb{E}_{\sigma \sim \texttt{RSD}}[\mathrm{SW}_0(\sigma, \tilde{\mu})]$.

We now bound the distortion of RRSD as follows.

$$\rho = \sup_{\mu} \frac{\mathrm{SW}(\pi^\star, \mu)}{\mathbb{E}_{\pi \sim \texttt{RRSD}}[\mathrm{SW}(\pi, \mu)]} \leq \sup_{\tilde{\mu}} \frac{T \cdot \mathrm{SW}_0(\sigma^\star, \tilde{\mu})}{T\mathbb{E}_{\sigma \sim \texttt{RSD}}[\mathrm{SW}_0(\sigma, \tilde{\mu})]} = \sup_{\tilde{\mu}} \frac{\mathrm{SW}_0(\sigma^\star, \tilde{\mu})}{\mathbb{E}_{\sigma \sim \texttt{RSD}}[\mathrm{SW}_0(\sigma, \tilde{\mu})]}$$

Since the last quantity is just the distortion of RSD under a one-shot matching problem, we can apply Lemma 4 from [19] and get a bound of $O(\sqrt{S})$ on the distortion. □

A.6 Proof of Theorem 5

Proof. We will write i_j to denote agent i-s j-th favourite item, and $\mu_{i,j}$ to denote the corresponding mean reward. By Lemma 2, agent i gets her j-th favourite item in at least $\frac{P}{2N}$ groups. Within any such group, there are T/P time slots, and agent i is assigned her j-th favourite arm for at least $\left\lfloor T/(PD_{i,i_j}) \right\rfloor$ times. Therefore, the total welfare guaranteed by DRRSD is at least

$$\sum_{i=1}^{N}\sum_{j=1}^{S} \mu_{i,j} \frac{P}{2N} \left\lfloor \frac{T}{PD_{i,i_j}} \right\rfloor \geq \sum_{i=1}^{N}\sum_{j=1}^{S} \mu_{i,j} \frac{P}{4N} \frac{T}{PD_{i,i_j}} = \frac{T}{4N} \sum_{i=1}^{N}\sum_{j=1}^{S} \frac{\mu_{i,j}}{D_{i,i_j}}$$

On the other hand, consider a matching algorithm that assign item i_j to agent i exactly $A_{i,j}$ times. Whenever item j is matched to agent i, it is blocked for D_{i,i_j} rounds. This implies that $A_{i,j} \leq T/D_{i,i_j}$. Therefore, the maximum welfare achievable by such a matching algorithm is at most

$$\sum_{i=1}^{N}\sum_{j=1}^{S} \mu_{i,j} A_{i,j} \leq \sum_{i=1}^{N}\sum_{j=1}^{S} \mu_{i,j} \frac{T}{D_{i,i_j}}$$

This establishes that the distortion of DRRSD is at most $4N = O(S)$. □

B Proofs for Section 4

B.1 Proof of Theorem 6

We restate the three claims of the theorem below:
(i) The expected dynamic $1/\sqrt{S}$-regret of the central mechanism is at most $O(\tilde{D}\sum_i H_i \log(NT))$.
(ii) For each agent i, reporting truthfully after the exploration phase would yield an expected $(1-1/e)$-IC regret of $O(H_i \log(NT))$.
(iii) Using async-MATLR, the number of time steps needed in the exploration phase is at most twice as many as the optimal allocation requires.

Proof. We consider a more general case where we do not have any further assumptions on BASE_i. Claim (i) can be proved as follows. By assumption, we know that each BASE_i would need an expected number of $H_i \log(1/\delta)$ pulls to learn the correct ranking with at least $(1-\delta)$ probability. By setting $\delta = \frac{1}{NT}$ for all BASE_i, we have that in expectation, the total number of pulls required by the agents is $\sum_i H_i \log(NT)$. As pulling an arm would come with a blocking delay of $D_{i,j} \leq \tilde{D}$ (where the (i,j) pair denotes that agent i pulls arm j), the total time needed to schedule these $\sum_i H_i \log(NT)$ pulls is at most $\tilde{D}\sum_i H_i \log(NT)$. Note that since we do not have any further information on BASE_i, it is typically not possible to further improve this bound.

Now we turn to the second phase. Using the union bound, we can easily show that after MATRL, each agent will learn their correct ranking of the arms with at least $(1-1/T)$ probability. Unless explicitly stated, we will consider this case (i.e., all the agents have their correct ranking). For the sake of simplicity, let OPT denote the social welfare of the optimal offline matching $M_T^*(\mu, D)$ (i.e., $\text{OPT} = \text{SW}(M_T^*(\mu, D))$. In addition, let OPT_2 denote the the social welfare of the optimal offline matching for the remaining time steps in the second phase. As each reward is bounded between $[0,1]$, it is easy to show that

$$\mathbb{E}\left[\text{OPT}_2\right] \geq \text{OPT} - \tilde{D}\sum_i H_i \log(NT) \tag{3}$$

where the expectation is on the length of the MALTR phase. This implies that the performance of RRSD, denoted by SW(RRSD), can be bounded below as follows:

$$\mathbb{E}\left[\text{SW(RRSD)}\right] \geq \frac{1}{\sqrt{S}}\text{OPT} - \frac{1}{\sqrt{S}}\tilde{D}\sum_i H_i \log(NT). \tag{4}$$

For the case when at least 1 agent does not learn the correct ranking, the regret suffered by RRSD is bounded above by NT. But this case only occurs with at most $1/T$ probability. Combining these 2 cases together, we have that the (dynamic approximate) $1/\sqrt{S}$-regret of RRSD can be bounded above by $\frac{1}{\sqrt{S}}\tilde{D}\sum_i H_i \log(NT) + N$.

To prove claim (ii), note that due to RRSD, if an agent changes its own preference report O_i, then it can only affect its own sequence of pulls. Therefore, by following a similar argument as above, and replacing the distortion bound with the incentive ratio bound, we get that the $1-1/e$-IC regret of each agent is $\frac{1}{\sqrt{S}}\tilde{D}\sum_i H_i \log(NT) + 1$.

Finally, to prove claim (iii), we consider the following scheduling problem:

Open Shop Scheduling: An instance of the open shop problem consists of a set of N machines and S jobs. Associated with each job s is a set of N independent tasks $s_1, \ldots, s_N$. The task s_k for job s must be processed on machine k for an uninterrupted $D_{s,k}$ time units. A schedule assigns

every task s_k to a time interval $D_{s,k}$ so that no job is simultaneously processed on two different machines, and so that no machine simultaneously processes two different jobs. The makespan C_{max} of a schedule is the longest job completion time. The optimal makespan is denoted by C^*_{max}.

It is easy to show that in our case, we need to schedule the pulls within the MALTR phase. As it can be shown that a greedy algorithm is a 2-approximation algorithm for the Open Shop Scheduling problem [46], we just need to observe that async-MATLR is that greedy algorithm. □

C Computational complexity of BLOCKMATCH

In this section, we investigate the computational complexity of the offline BLOCKMATCH problem. Observe that, when there is only one agent, BLOCKMATCH corresponds to the stochastic blocking bandit problem defined in [10]. Therefore, offline BLOCKMATCH inherits the complexity issues of the offline stochastic blocking bandit problem. More precisely, even when the mean rewards are known to the central mechanism, there is no pseudopolynomial time algorithm for offline BLOCKMATCH unless the randomised exponential time hypothesis [15] is false. In their proof of this result for stochastic blocking bandits, [10] perform a reduction from the PINWHEEL SCHEDULING problem [21]. For the sake of clarity, we provide an essentially identical reduction for the offline BLOCKMATCH problem in the proof below.

Theorem 7. *Even if the central mechanism is aware of the reward profile of each agent, the offline BLOCKMATCH problem does not admit a pseudopolynomial time policy unless the randomised exponential time hypothesis is false.*

Proof. We show the hardness of the BLOCKMATCH problem by reducing from the PINWHEEL SCHEDULING problem defined as follows:

PINWHEEL SCHEDULING [21]: Given a set of tasks, $\mathcal{S} = \{1, \ldots, N\}$, with minimum repeat times, $\{d_j \in \mathbb{N} : j \in \mathcal{S}\}$, the PINWHEEL SCHEDULING problem is to decide whether there exists a schedule (i.e. a mapping $\Sigma : [T] \to \mathcal{S}$ for any $T \geq 1$) such that for each task $j \in \mathcal{S}$ and every consecutive sequence of time steps of length d_j, task j is performed at least once. Note that only one task can be performed per time step.

We call such a schedule, if it exists, a valid schedule. A PINWHEEL SCHEDULING instance with a valid schedule is a YES instance, ohterwise it is a NO instance. A PINWHEEL SCHEDULING instance is called dense if $\sum_{j=1}^{N} \frac{1}{d_j} = 1$.

Given a dense instance of a PINWHEEL scheduling problem we construct an instance of the offline BLOCKMATCH problem in the following manner. We consider an offline BLOCKMATCH instance with a single agent and a set of $N+1$ services. For each task j we associate a service j and set the mean reward of all such services to $\frac{1}{N}$. The blocking delay of each service is set equal to d_j, the minimum repeat time of the associated task. Additionally, we include one more service, with mean reward equal to zero, and no blocking delay.

Case 1: PINWHEEL SCHEDULING instance is a YES instance, i.e. there exists a valid schedule. Furthermore, as the instance is dense, each task must appear with exact period d_j in the schedule. Additionally, for all $T \geq 1$, there are no empty time slots in the schedule. This implies that assigning the agent services according to the valid schedule is an optimal policy according to the BLOCKMATCH problem with cumulative reward T/N in in T time slots, for any $T \geq 1$.

Case 2: PINWHEEL SCHEDULING instance is a NO instance, i.e. there does not exist a valid schedule. This implies that, for any schedule, there exists a block of d_j time slots such that task j is not performed in that block, for some $j \in \mathcal{S}$. As the instance is dense, this implies that there

must exist a gap in any schedule. In turn, this implies that in the offline BLOCKMATCH problem, any valid policy must assign the agent service $N+1$ at least once. Moreover any valid schedule is periodic with period at most $\Pi_{j=1}^{N} d_j$. Together, these facts imply that for $T \geq 1$ we can achieve at most $\left(T - \lfloor \Pi_{j=1}^{N} d_j \rfloor\right)$ welfare.

Thus, for large enough T, there is a non-zero gap in the welfare obtained in both cases. Therefore by solving the given BLOCKMATCH instance, we can determine whether the PINWHEEL SCHEDULING instance is a YES instance or a NO instance. The result then follows from Theorem 3.1 and Theorem 24 of [25]. More specifically, in [25] it is shown that PINWHEEL SCHEDULING with dense instances does not admit any pseduopolynomial time algorithm unless the randomised exponential time hypothesis is false. □

D Updates and Corrections

- The pseudocode for `RRSD` has been updated. More precisely, the output line was changed to use the correct notation and single line comments have been updated to match the formatting of multi-line comments. In addition, a few simple comments have been added.
- The big O notation was updated to be made consistent in each theorem statement.
- References were added for PINWHEEL SCHEDULING [21, 25] and the randomised exponential time hypothesis [15].