Sequential Blocked Matching

Nicholas Bishop, [1] Hau Chan, [2] Debmalya Mandal, [3] Long Tran-Thanh [4]

[1]University of Southampton, [2] University of Nebraska-Lincoln, [3] Max Planck Institute, [4] University of Warwick
nb8g13@soton.ac.uk, hchan3@unl.edu, dmandal@mpi-sws.org, long.tran-thanh@warwick.ac.uk

Abstract

We consider a sequential blocked matching (SBM) model where strategic agents repeatedly report ordinal preferences over a set of services to a central planner. The planner'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 strategic agents are aware of their 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 random serial dictatorship which match our 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 initially unaware of her true preferences, and the planner must facilitate each agent in the learning of their preferences 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. Motivated by such constraints in multi-agent systems, 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 their ordinal preferences based on their expected rewards over a set of alternatives, or *services*. The agents are matched to the services 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 (Durlauf and Blume 2008; Roth, Sönmez, and Ünver 2004), assigning students to rooms in residential halls (Durlauf and Blume 2008), allocating workers to tasks in crowdsourcing (Difallah, Demartini, and Cudré-Mauroux 2013; Aldhahri, Shandilya, and Shiva 2015), and recommending users to activities in recommender systems (Satzger, Endres, and Kießling 2006; Ansari, Essegaier, and Kohli 2000; Isinkaye, Folajimi, and Ojokoh 2015).

In many of these situations, there are several obstacles. First, for a setting with reusable 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, and the planner 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 (Waldfogel 2008). Note that, even when the agents are themselves using learning algorithms over time, coordination by the planner becomes necessary to avoid different agents exploring the same service at a time – a problem which is exacerbated by blocking of the services.

Finally, in several settings, the agents are aware of their preferences, but they might be *strategic* in reporting their preferences for getting 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 matching outcomes.

1.1 Main Contributions

In order to capture the notion of one-sided matching with blocking, we introduce a *sequential blocked matching* (SBM) model, in which a set of n strategic agents are matched to a set of s services repeatedly over rounds and where matched services are *blocked* for a deterministic number of time steps. Each agent reports her ordinal preferences over the services every round, based on her current estimates of the expected rewards of the services. As is standard in the matching literature, we focus on the setting where agents just report ordinal preferences over the services. The planner's goal is to derive a matching policy that, at each round, elicits true preferences from the agents and matches them to services in order to maximize the expected social welfare, which is the sum of the expected utilities of the agents from the matchings over time, whilst accounting for the blocking of services. 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.

We investigate the offline and online variations of the SBM model. For both variations, we are interested in deriving deterministic and randomized policies that are approximately truthful and efficient. We measure truthfulness by *incentive ratio* (Chen et al. 2012), which measures how much a single agent can gain via misreporting preferences. We measure efficiency through the notion of *distortion* from social choice theory (Procaccia and Rosenschein 2006), 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. In the offline setting of SBM, each agent knows their own preferences and rewards over the services, but the planner does not. In addition, each agent reports their preferences only once to the planner, before matching begins. 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 random serial dictatorship (RSD) mechanism 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 or rewards and must learn their preferences via repeated matching to services. After each matching, the agents update their preferences and strategically report them to the planner. We design an approximately truthful mechanism, bandit RRSD (BRRSD), based on the Explore-then-Commit (ETC) paradigm, which achieves sublinear dynamic approximate regret. In particular, BRRSD has two phases. In the first phase, it allows the participating agents to learn their preferences via uniform allocation of services. Using the learnt estimates from this phase, the mechanism then runs RRSD in the second phase.

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.

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* (Procaccia and Rosenschein 2006) 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 (Boutilier et al. 2015; Mandal et al. 2019; Anshelevich et al. 2018; Kempe 2020; Anshelevich and Postl 2017). Our offline benchmark is related to the literature on the distortion of matching (Amanatidis et al. 2021; Filos-Ratsikas, Frederiksen, and Zhang 2014; Anshelevich and Sekar 2016). However, our offline benchmark needs to consider repeated matching over time, and because of the blocking of services, 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., (Karp, Vazirani, and Vazirani 1990; Kalyanasundaram and Pruhs 1993; Karande, Mehta, and Tripathi 2011)) and stable matching (e.g., (Khuller, Mitchell, and Vazirani 1994)) where the matching entities (i.e. 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 agent preferences that can change over time (e.g., (Parkes and Procaccia 2013; Hosseini, Larson, and Cohen 2015a,b)). However, they do not consider the problem of maximizing expected rewards and blocking.

Blocking Bandits. Our work in the online SBM models is closely related to the recent literature on blocking bandit models (Basu et al. 2021, 2019; Bishop et al. 2020), where each pulled arm (i.e., service) can be blocked for a fixed number of rounds. Our work is also related to bandits with different types of arm-availability constraints (Neu

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*)	$O(s)$	$(1 - 1/e)$
Any Randomised Mechanisms (*lower bound*)	$\Omega(\sqrt{s})$	$(0, 1]$
Repeated Random Serial Dictatorship (*upper bound*)	$O(\sqrt{s})$	$(1 - 1/e)$

and Valko 2014; Kleinberg and Immorlica 2018; Kleinberg, Niculescu-Mizil, and Sharma 2010). 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 a planner in order to be matched to some arm at each round.

Multi-agent multi-armed bandits. The online setting in our work is broadly related to the growing literature on multi-agent multi-armed bandits (Liu, Mania, and Jordan 2020; Sankararaman, Basu, and Sankararaman 2021; Bistritz et al. 2020). Liu, Mania, and Jordan (2020) consider a matching setting where strategic agents learn their preferences over time, and the planner 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 services, whereas the existing works compete against a fixed benchmark e.g. repeated applications of Gale-Shapley matching in each round (Liu, Mania, and Jordan 2020).

2 Preliminaries

In this section, we introduce our model for sequential blocked matching. We start by describing how the preferences of each agent are modeled and describe formally how agents can be matched to services in a single time step. After which, we introduce the first of two sequential settings that we study in this paper, which features the blocking of services when they are assigned to agents.

In our model, we have a set of agents, $N = \{1, \dots, n\}$, who hold *cardinal* preferences over a set of services, $S = \{1, \dots, s\}$, where $s \gg n$ [1]. We use $\mu_{i,j} \in \mathbb{R}^+$ to describe the cardinal reward agent i receives for being assigned service j. Similarly, we denote by $\mu_i = (\mu)_{j=1}^s$ the vector of 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 $\mu_i \in \Delta^{s-1}$ for all $i \in N$. 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 (Filos-Ratsikas, Frederiksen, and Zhang 2014), and are typically required in order to prove lower bounds for truthful algorithms such as RSD. Moreover, the unit-sum assumption is prevalent in social choice theory (Boutilier et al. 2015). Lastly, we denote by μ the n by s matrix of rewards.

We say that agent i (weakly) prefers service a to service b if agent i receives greater reward by being assigned service a over service b. That is, agent i prefers service a over service b if and only if $\mu_{i,a} \geq \mu_{i,b}$. We use the standard notation $a \succ_i b$ to say that agent i prefers service a to service b. Additionally, we use the notation $\succ_i(j)$ to indicate the service in the jth position of the preference ordering $\succ_i$. Note that every reward profile induces a linear preference ordering of services [2]. We use the notation $\mu_i \rhd \succ_i$ to denote that $\succ_i$ is a preference ordering induced by agent i's reward profile. We let $\mathcal{P}(S)$, or $\mathcal{P}$ for short, denote the class of all linear preferences over S. We write $\succ_i$ to denote the preferences induced by agent i's reward profile. Furthermore, we let $\succ = (\succ)_{i=1}^n \in \mathcal{P}^n$ denote the *preference profile* of the agents. As is standard, we write $\succ_{-i}$ to denote $(\succ_1, \dots, \succ_{i-1}, \succ_{i+1}, \dots, \succ_n)$. As a result, we may denote $\succ$ by $(\succ_i, \succ_{-i})$.

A *matching* $m : N \to S \cup \{0\}$ is a mapping from agents to services. We let $m(i)$ denote the service allocated to agent i by the matching m. We use 0 to denote the null assignment. That is, agent i is assigned no service in a matching if $m(i) = 0$. We let $\emptyset$ denote the null matching, in which no agent is assigned a service. We say matching is *feasible* if no two agents are mapped to the same service. We let $\mathcal{M}$ denote the set of all feasible matchings.

In this paper, we consider discrete-time sequential decision problems, in which a planner selects a sequence of (feasible) matchings over T time steps. We let m_t denote the matching chosen by the planner at time step t, and denote by $M = (m_t)_{t=1}^T$ a sequence of T matchings. We denote by $M(t, i) = m_t(i)$ the service matched to agent i at time t.

Furthermore, we assume that, when a service is assigned, it may be blocked for a time period depending on the agent it was assigned 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. We 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 blocking delays.

From now on, we assume that all blocking delays are known a priori by both the planner and all 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*

[1] Note that this is without loss of generality, as we may always add dummy services corresponding to a null assignment.

[2] One reward profile may induce many linear orderings. However, the linear preference profile induced by a reward profile can be made unique via tie-breaking rules.

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 $t \in \{1, \ldots, T\}$, $i \in N$, and $j \in S$, if $M(t, i) = j$ then $M(t', i') \neq j$ for all $i' \in N$ and for all t' such that $t < t' \leq t + D_{i,j} - 1$.

In other words, we say that a matching sequence is feasible if there is no matching in the sequence which assigns an agent a service which has been blocked by a previous matching. 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.

We define the *utility*, $\mathrm{W}_i(M, \mu_i)$, agent i receives from a matching sequence M as the sum of rewards it receives from each matching in the sequence. That is, $\mathrm{W}_i(M, \mu_i) = \sum_{t=1}^{T} \mu_{i,M(t,i)}$. Similarly, we define the *social welfare*, $\mathrm{SW}(M, \mu)$, of a matching sequence M as the summation of the utilities for all agents. More specifically, $\mathrm{SW}(M, \mu) = \sum_{i=1}^{n} W_i(M, \mu_i)$.

Next, we will describe the first sequential matching setting we consider in this paper, which we call the offline SBM setting. In this setting, the planner must produce a feasible matching sequence of length T. Prior to the selection of a matching sequence, each agent submits a linear preference ordering to the planner. We denote by $\tilde{\succ}_i$ the preference ordering, or *report*, submitted by agent i. Analogously, we define $\tilde{\succ}$ as the preference profile submitted cumulatively by the agents, and call it the *report profile*. A *matching policy* $\pi(M \mid \tilde{\succ}, D)$ assigns a probability of returning a matching sequence M given a submitted report profile $\tilde{\succ}$ and blocking delay matrix D. When it is clear from context, we will abuse notation and use $\pi(\tilde{\succ}, D)$ to refer to the (random) matching sequence prescribed by a policy π given a report profile $\tilde{\succ}$ and blocking delay matrix D.

We say that a matching policy is *admissible*, if for all possible report profiles and blocking delay matrices, the matching sequence returned by the policy is always feasible. The goal of the planner is to adopt an admissible matching policy which achieves high social welfare in expectation relative to the best feasible matching sequence in hindsight, $M^*(\mu, D) = \mathrm{argmax}_{M \in \mathcal{M}_T^D} \mathrm{SW}(M, \mu)$.

We assume that each agent, with full knowledge of the matching policy employed the planner, submits a linear preference ordering with the intention of maximising their own utility, and therefore may try to manipulate the planner by submitting a preference ordering which is not induced by their underlying cardinal preferences. We say that an agent is *truthful* if they submit a preference ordering induced by their underlying cardinal preferences. That is, an agent is truthful if $\mu_i \rhd \tilde{\succ}_i$. We denote by $\succ_i^*$ the report by agent i which maximises agent i's utility in expectation under the assumption that all other agents are truthful. We say that a policy is *truthful* if for all possible μ and D it is optimal for each agent to be truthful if all other agents are truthful. In other words, a policy is truthful if for all μ and D we have that $\mu_i \rhd \succ_i^*$ for all $i \in N$.

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

Definition 2. *The distortion of a matching policy is the worst-case ratio between the expected social welfare of the matching sequence, $\pi(\succ, D)$, returned by the policy under the assumption that all agents are truthful, and the social welfare of the optimal matching sequence, $M^*(\mu, D)$:*

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

Definition. *The distortion of a matching policy π is the worst-case ratio between the expected social welfare of the matching sequence, $\pi(\succ, D)$, returned by π and the social welfare of the optimal matching sequence, $M^*(\mu, D)$:*

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

Note that distortion is the approximation ratio of the policy π with respect to best matching sequence. In addition, note that the distortion is only a useful measure of a matching policies efficiency if said policy encourages truthful reporting. For example, for truthful policies, distortion is completely characterising of a policy's expected performance. As a result, we not only seek policies which have low distortion, but also policies which incentivise agents to submit their reports truthfully.

To this end, we introduce the notion of incentive ratio, which measures the relative improvement in utility an agent can achieve by lying about their preferences.

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

$$\zeta(\pi) = \max_{D, \succ_{-i}, \mu_i \rhd \succ_i} \frac{\mathbb{E}[W_i(\pi((\succ_i, \succ_{-i}), D), \mu_i)]}{\mathbb{E}[W_i(\pi((\succ_i^*, \succ_{-i}), D), \mu_i)]}$$

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

$$\zeta(\pi) = \max_{D, \succ_{-i}, \mu_i \rhd \succ_i} \frac{\mathbb{E}[W_i(\pi((\succ_i, \succ_{-i}), D), \mu_i)]}{\mathbb{E}[W_i(\pi((\succ_i^*, \succ_{-i}), D), \mu_i)]}$$

If a policy has an incentive ratio of 1, then it is truthful. There are many reasons that we may expect a policy with bounded incentive ratio to do well. A bounded incentive ratio implies truth telling is a good approximation to the optimal report. If computing the optimal report is computationally intractable for the agent, being truthful is therefore an attractive alternative, especially if the approximation ratio implied by the incentive ratio is tight. In summary, we seek matching policies with good guarantees when it comes to both incentive ratio and distortion. This topic is treated in detail in the forthcoming sections.

3 The Offline SBM Setting

In this section, we present our analysis of the offline SBM setting. We first provide a lower bound on the distortion achievable by both randomised and deterministic policies. Then, we discuss why trivial extensions of truthful one-shot matching algorithms do not result in truthful policies. Instead, we focus on designing policies which use truthful one-shot matching mechanisms as a basis, and have bounded incentive ratio. More precisely, we present the RRSD algorithm. We show that the incentive ratio of RRSD is bounded below by $1-1/e$, and provide upper bounds on the distortion achieved by RRSD, which match our previously established lower bounds on the best distortion achievable by any randomised algorithm.

3.1 Lower Bounds on the Distortion of Deterministic and Randomised Policies

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, we show that there exists a set of reward profiles which induces the aforementioned ordinal preferences and on which M incurs distortion of order $\Omega(s)$.[3]

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

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. That is, policies that assign each service to each agent the same number of times in expectation for all possible preference profiles. Then, we construct a set of reward profiles which yields the desired distortion for all anonymous truthful policies.

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

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

3.2 Constructing Truthful Algorithms for the Offline SBM Setting

As previously mentioned, we assume that agents submit reports with the intention of maximising their own utility. As a result, the distortion incurred by a policy may not reflect its performance in practice, as agents may be incentivised to misreport their preferences in order to increase their utility. Note that in standard one-shot one-sided matching problems, this issue is sidestepped via the employment of truthful policies, like RSD. In addition, the restriction to considering truthful policies is well justified by the revelation principle. In a similar way, we would like to develop truthful algorithms for the offline SBM setting.

One may be tempted to apply such truthful one-shot policies to our setting directly. That is, to apply an algorithm such as RSD repeatedly on every time step in sequence in order to devise a matching sequence. This intuition is correct when there is no blocking, as the matching problems for each time step are then independent of each other. However, with blocking, the matchings from previous time steps will have a substantial effect on the set of matchings which preserve the feasibility of the matching sequence in future rounds. As a result, immediately obvious approaches, such as matching according to RSD repeatedly, do not result in truthful policies.

One simple way of generating truthful policies is to run a truthful one-shot one-sided matching policy once every $\tilde{D}$ time steps and simply return the empty matching 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 SBM setting 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 bounded incentive ratio.

3.3 A Greedy Algorithm for the Offline SBM Setting

The RRSD algorithm slowly builds up a matching sequence M over time by iterating through agents and services. In other words, RRSD begins with the empty matching sequence, where $M(t,i) = 0$ for all t and $i \in N$. To begin, RRSD samples a permutation of agents σ uniformly at random. Next, RRSD iterates through the agents in the order given by the permutation sampled. For each agent i, RRSD iterates through services in the order specified by the preference ordering $\tilde{\succ}_i$ reported by agent i. For a given service j, RRSD repeatedly assigns service j to agent i at the earliest time step which does not cause the matching sequence to become infeasible. When no such time step exists, RRSD moves onto the next service in agent i's preference ordering. Once RRSD has iterated through the entire preference ordering of agent i, RRSD moves onto the next agent in the permutation σ and repeats this process until the end of the permutation is reached. The pseudocode for RRSD is given in the full version.

We will now briefly give the intuition behind RRSD. In essence, RRSD attempts to mimic the RSD algorithm for one-shot matching problems by allowing each agent to sequentially choose a feasible assignment of services over the entire time horizon (whilst respecting the assignments chosen by previous agents) via its reported ordering. In the case of RSD, given an agent's preference ordering, the same assignment is always optimal no matter the underlying the reward profile of the agent. That is, it is optimal for the agent to be assigned its most preferred available service, no matter its cardinal preferences. As a result, RSD is trivially truthful in the one-shot matching setting. In contrast, in the offline SBM setting, the optimal assignment of services can be dif-

[3] All missing proofs are deferred to the full version

ferent for two reward profiles which induce the same preference ordering. Hence, there is no trivial assignment, based on the preference ordering submitted by the agent which guarantees that agents are truthful.

Instead, given an agent's preference ordering, we attempt to find an assignment which performs reasonably well, no matter the underlying reward profile of the agent. RRSD uses a greedy algorithm to compute the assignment given to an agent. As long as this greedy algorithm is a good approximation of the optimal assignment, no matter the agent's underlying reward profile, then RRSD will have a bounded incentive ratio. The next theorem formalises this argument.

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

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

Remark. It is an open question as to whether we can achieve incentive ratios better than $1 - 1/e$ when RRSD is used. In particular, one can show that many scheduling problems such as generic job interval scheduling and (dense) pinwheel scheduling can be reduced to the optimal manipulation problem each agent faces in RRSD. Whilst it is known that generic job interval scheduling problems are MAXSNP-hard (Chuzhoy, Ostrovsky, and Rabani 2006), it is still not known whether there exists a scheduling algorithm with approximation ratio better than $1 - 1/e$.

We now provide an upper bound on the distortion achieved by RRSD, which matches our previously established lower bound for randomised policies described in Theorem 2.

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

Finally, we show that it is possible to match the previously established lower bound for the distortion of deterministic algorithms. More specifically, we show that a derandomised version of RRSD incurs a distortion of at most $O(s)$. The main idea is that we can select $O(n^2 \log n)$ permutations of agents so that the number of times an agent i is selected in the jth position is $\Theta(n \log n)$. We can then run through these permutations one by one instead of selecting one permutation uniformly at random as in the RRSD algorithm.

Theorem 5. *There is an admissible deterministic policy with distortion at most $O(s)$ for any $T \geq O(n^2 \log(n))$*

4 SBM with Bandit Feedback

Note that, in order for the guarantees above to hold in practice, we must assume that agents are fully aware of their ordinal preferences before matching begins. However, in many real-world scenarios, agents may be initially unaware of their preferences and learn them over time by matching with services. In addition, the reward an agent receives for being matched with a service may be inherently stochastic, depending on unobservable aspects of the underlying environment. With these concerns in mind we present a new sequential blocked matching setting, which we call the online SBM setting with bandit feedback, or online SBM for short.

In the online SBM setting, matching occurs time step by time step. At the beginning of each time step, every agent must submit a report, $\tilde{\succ}_i^t$, to the planner. The planner is then tasked with returning a matching of agents to services which obeys the blocking constraints imposed by the matchings from previous time steps. At the end of each time step, agent i receives a reward, $r_{i,t} \in [0, 1]$, sampled from a distribution with mean $\mu_{i,j}$, where j is the service agent i was assigned in the matching returned by the planner. Additionally, we assume that each agent maintains an internal estimation, $\succ_i^t$, of its own preference ordering at every time step, based on the rewards received thus far.

We use $H_t^{\succ} = (\tilde{\succ}^1, \dots, \tilde{\succ}^t)$ to denote the *report history* up to time step t. Furthermore, we use $H_t^m = (m_1, \dots, m_t)$ to describe the *matching history* at the end of time step t. We say that a matching history is *feasible* if its matchings form a feasible matching sequence. Similarly, we use $H_t^r = (r_1, \dots, r_t)$ to denote the *reward history*. That, is the tuple of reward vectors, r_t, observed by the agents at every time step. An *(online) matching policy* $\pi = (\pi_1, \dots, \pi_T)$ is a tuple of functions $\pi_t(m|\tilde{H}_t^{\succ}, H_t^m, D)$ which assigns a probability of returning the matching m given a report history $H_t^{\succ}$, a feasible matching history H_t^m and a blocking delay matrix D. Similarly to the offline setting, we say that a matching policy is *admissible* if it always returns a feasible matching sequence.

Likewise, an *(online) report policy* for agent i, $\tilde{\psi}_i = (\tilde{\psi}_1, \dots \tilde{\psi}_t)$, is a tuple of functions $\tilde{\psi}_t(\tilde{\succ}_i^t | H_t^r, H_t^m, D)$ which assign a probability of agent i reporting $\tilde{\succ}_i^t$ at time step t given a reward history H_t^r, a matching history H_t^m, and blocking delay matrix D. We denote by $\tilde{\psi} = (\tilde{\psi}_1, \dots, \dots \tilde{\psi}_n)$ the tuple of report policies used by the agents. As before we use the notation $\tilde{\psi}_{-i}$ to denote the report policies of all agents bar agent i and use ψ to denote the tuple of report policies where each agent reports its internal estimation $\succ_i^t$ at every time step. We say that an agent is *truthful* if it employs the report policy ψ_i.

The goal of each agent is to employ a report policy that maximises the sum of their own rewards across the time horizon. In contrast, goal of the planner is to employ a matching policy which maximises the sum of rewards across all agents and across all time steps.

In the bandit literature, a performance metric that is typically used to measure the efficiency of a policy is regret, which is defined as the expected difference between the rewards accumulated by a matching policy, and the expected reward accumulated by the best fixed matching policy in hindsight. That is, the best policy which repeatedly selects the same matching in as many time steps as possible. Such a benchmark policy may have very poor performance relative to the optimal matching sequence in expectation, and as such, the classical notion of regret is an unsuitable performance measure in the online SBM setting. To resolve this issue, we propose the following regret definition:

Definition 4. *The dynamic α-regret of a policy π is:*

$$R_{\pi}^{\alpha}(D, \mu, T) = \alpha SW(M^*, \mu) - \mathbb{E}_{\psi,\pi}\left[\sum_{i=1}^{n}\sum_{t=1}^{T} r_{i,t}\right]$$

In other words, we compare the expected performance of a matching policy against a dynamic oracle which returns an $1/\alpha$-optimal solution to the corresponding offline SBM problem, under the assumption that agents truthfully report their internal estimation of their preferences at each time step. Recall that, in the offline SBM setting, the distortion incurred by any policy is at least $\Omega(s)$. As a result, we cannot expect to construct algorithms with vanishing $1/\alpha$-regret for $\alpha < \sqrt{s}$. In addition, one would not expect any matching policy to have low dynamic regret if the internal estimations computed by each agent are inaccurate. For example, if any agent's internal estimator consists of returning a random preference ordering, then we cannot hope to learn about said agent's preferences. As a result, we need to make reasonable assumptions regarding the internal estimator of each agent.

Similar to distortion for the offline SBM setting, dynamic α-regret is only a meaningful performance measure for policies which motivate agents to adopt truthful reporting policies. Inspired by the concept of incentive ratio for the offline SBM setting, we define a new notion of regret which, given a matching policy π captures the expected gain in cumulative reward an agent can achieve by misreporting.

Definition 5. *For a given matching policy π, we define agent i's α–IC regret (or α incentive compatible regret) as follows:*

$$I^{\alpha}_{\pi}(D,\mu,T) = \alpha \max_{\tilde{\psi}} \mathbb{E}_{\left(\psi_{-i},\tilde{\psi}_i\right),\pi}\left[\sum_{t=1}^{T} r_{i,t}\right] - \mathbb{E}_{\psi,\pi}\left[\sum_{t=1}^{T} r_{i,t}\right]$$

Note that for some matching policies, computing the optimal reporting policy may be computational intractable. If agents have vanishing α-IC regret for a such policy, then adopting a truthtelling forms a good approximation of each agent's optimal reporting policy. If this approximation is better than what can be computed by the agent, then we can expect each agent to adopt their truthful reporting policy. Thus, we seek matching policies with good guarantees with respect to both dynamic α-regret and α-IC regret.

4.1 Algorithms for Online SBM

Next, we present a matching policy which achieves meaningful guarantees with respect to both dynamic α-regret and α-IC regret. More precisely, we present the bandit repeated random serial dictatorship (BRRSD) algorithm. Before we describe BRRSD formally, we first state our assumptions regarding the internal estimator used by each agent.

Let $\hat{\mu}_{i,j}$ denote the empirical mean of the reward samples agent i receives from being assigned service j. We say that an agent i is *mean-based* if service a is preferred to service b in $\succ_i^t$ if and only if $\hat{\mu}_{i,a} \geq \hat{\mu}_{i,b}$. That is, a mean-based agent prefers services with higher empirical mean reward. From hereon, we assume that all agents are mean-based.

Additionally, we use Δ_{min} to denote the smallest gap in mean rewards between two services for the same agent. That is, $\Delta_{\text{min}} = \min_{i,a\neq b} |\mu_{i,a} - \mu_{i,b}|$. Note that Δ_{min} is analogous to common complexity measures used in bandit exploration problems. Intuitively, if the mean rewards received from being assigned two services are similar, it will take more samples for a mean-based agent to decide which service they prefer.

We are now ready to describe BRRSD. BRRSD is split into two phases. In the first phase, BRRSD assigns each agent each service exactly $\lceil 2\log(2Tsn)/\Delta^2_{\text{min}} \rceil$ times. BRRSD performs these assignments in a greedy manner. At each time step, BRRSD iterates through the agent-service pairs that still need to be assigned in an arbitrary order. If an agent-service pair does not violate blocking constraints, then it is added to the current matching. Once this iteration is completed, or all agents have been assigned services, the matching is returned and BRRSD moves onto the next time step. Once all required assignments have been completed, BRRSD waits until all services are available, matching no agents to services in the meantime. Note that this takes a maximum of $\tilde{D}$ rounds. Then, BRRSD begins its second phase. At the beginning of the next time step, BRRSD observes the report profile $\tilde{\succ}_i^t$ and selects matchings according to RRSD using this report profile for the remainder of the time horizon. The full pseudocode for BRRSD is deferred to the full version.

BRRSD falls in the class of explore-then-commit (ETC) algorithms common in the bandit literature. The first phase of BRRSD serves as an exploration phase in which agent's learn their preference ordering. Meanwhile, the second phase of BRRSD serves as exploitation phase in which agents have the opportunity to disclose their accumulated knowledge to the planner in the form of ordinal preferences. Observe that this decoupling of exploration and exploitation avoids complicated incentive issues that may arise for sequential algorithms, which make no such clear separation.

The exploration phase of BRRSD is simple relative to typical approaches in the bandit exploration literature. One may hope to apply a more complicated scheme for exploration, however approaches with better performance guarantees typically depend directly on the reward samples observed, which the planner does not access to. The next theorem describes the guarantees of BRRSD.

Theorem 6. *Under the assumption that agents are mean-based, the following is true for all μ and D:*
(i) The dynamic $(1/\sqrt{s})$-regret of BRRSD is $O\left(\tilde{D}\sqrt{s}\log\left(Tsn\right)/\Delta^2_{min}\right)$.
(ii) The $(1-1/e)$-IC regret for all agents under BRRSD is $O\left(\tilde{D}s\log\left(Tsn\right)/\Delta^2_{min}\right)$.
(iii) The greedy algorithm used by BRRSD in the exploration phase uses at most twice as many time steps as the shortest feasible matching sequence which completes the required assignments.

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. For the offline setting, we lower bounded the performance of both deterministic and randomised policies, presented algorithms with matching performance guarantees and bounded incentive ratio. Then, we analysed an online SBM setting, in which agents are initially unaware of their preferences and must learn them. For this setting, we presented an algorithm with sublinear regret with respect to an offline approximation oracle.

There are many interesting directions for future work. A natural generalisation would be to consider a two-sided matching setting (Roth and Sotomayor 1992) where services also hold preferences over agents. Additionally, our algorithms for both the offline and online settings are centralised. It is worth investigating whether similar performance guarantees can be achieved by a decentralised approach. Furthermore, we assumed that the preferences are static over time. It remains to be seen whether our approach generalises to settings where agents' preferences are dynamic and change over time (Bergemann and Välimäki 2019).

Acknowledgements

Nicholas Bishop was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Partnership grant.

References

Aldhahri, E.; Shandilya, V.; and Shiva, S. 2015. Towards an Effective Crowdsourcing Recommendation System: A Survey of the State-of-the-Art. In *2015 IEEE Symposium on Service-Oriented System Engineering*, 372–377.

Amanatidis, G.; Birmpas, G.; Filos-Ratsikas, A.; and Voudouris, A. A. 2021. Peeking behind the ordinal curtain: Improving distortion via cardinal queries. *Artificial Intelligence*, 296: 103488.

Ansari, A.; Essegaier, S.; and Kohli, R. 2000. Internet Recommendation Systems. *Journal of Marketing Research*, 37(3): 363–375.

Anshelevich, E.; Bhardwaj, O.; Elkind, E.; Postl, J.; and Skowron, P. 2018. Approximating optimal social choice under metric preferences. *Artificial Intelligence*, 264: 27–51.

Anshelevich, E.; and Postl, J. 2017. Randomized social choice functions under metric preferences. *Journal of Artificial Intelligence Research*, 58: 797–827.

Anshelevich, E.; and Sekar, S. 2016. Blind, greedy, and random: Algorithms for matching and clustering using only ordinal information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Basu, S.; Papadigenopoulos, O.; Caramanis, C.; and Shakkottai, S. 2021. Contextual Blocking Bandits. In Banerjee, A.; and Fukumizu, K., eds., *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, 271–279.

Basu, S.; Sen, R.; Sanghavi, S.; and Shakkottai, S. 2019. Blocking Bandits. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32.

Bergemann, D.; and Välimäki, J. 2019. Dynamic mechanism design: An introduction. *Journal of Economic Literature*, 57(2): 235–74.

Bishop, N.; Chan, H.; Mandal, D.; and Tran-Thanh, L. 2020. Adversarial Blocking Bandits. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 8139–8149.

Bistritz, I.; Baharav, T.; Leshem, A.; and Bambos, N. 2020. My fair bandit: Distributed learning of max-min fairness with multi-player bandits. In *International Conference on Machine Learning*, 930–940. PMLR.

Boutilier, C.; Caragiannis, I.; Haber, S.; Lu, T.; Procaccia, A. D.; and Sheffet, O. 2015. Optimal social choice functions: A utilitarian view. *Artificial Intelligence*, 227: 190–213.

Calabro, C.; Impagliazzo, R.; Kabanets, V.; and Paturi, R. 2008. The complexity of Unique k-SAT: An Isolation Lemma for k-CNFs. *Journal of Computer and System Sciences*, 74(3): 386–393. Computational Complexity 2003.

Chen, N.; Deng, X.; Zhang, H.; and Zhang, J. 2012. Incentive Ratios of Fisher Markets. In Czumaj, A.; Mehlhorn, K.; Pitts, A.; and Wattenhofer, R., eds., *Automata, Languages, and Programming*, 464–475. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-31585-5.

Chuzhoy, J.; Ostrovsky, R.; and Rabani, Y. 2006. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4): 730–738.

Difallah, D. E.; Demartini, G.; and Cudré-Mauroux, P. 2013. 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*, 367–374.

Durlauf, S. N.; and Blume, L. 2008. *The new Palgrave dictionary of economics*. Palgrave Macmillan.

Filos-Ratsikas, A.; Frederiksen, S. K. S.; and Zhang, J. 2014. Social welfare in one-sided matchings: Random priority and beyond. In *International Symposium on Algorithmic Game Theory*, 1–12. Springer.

Hosseini, H.; Larson, K.; and Cohen, R. 2015a. Matching with Dynamic Ordinal Preferences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

Hosseini, H.; Larson, K.; and Cohen, R. 2015b. On Manipulablity of Random Serial Dictatorship in Sequential Matching with Dynamic Preferences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

Isinkaye, F.; Folajimi, Y.; and Ojokoh, B. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3): 261–273.

Kalyanasundaram, B.; and Pruhs, K. 1993. Online Weighted Matching. *Journal of Algorithms*, 14(3): 478–488.

Karande, C.; Mehta, A.; and Tripathi, P. 2011. Online Bipartite Matching with Unknown Distributions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, 587–596.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An Optimal Algorithm for On-Line Bipartite Matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, 352–358.

Kempe, D. 2020. Communication, distortion, and randomness in metric voting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2087–2094.

Khuller, S.; Mitchell, S. G.; and Vazirani, V. V. 1994. Online algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2): 255–267.

Kleinberg, R.; and Immorlica, N. 2018. Recharging bandits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 309–319. IEEE.

Kleinberg, R.; Niculescu-Mizil, A.; and Sharma, Y. 2010. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2): 245–272.

Liu, L. T.; Mania, H.; and Jordan, M. 2020. Competing bandits in matching markets. In *International Conference on Artificial Intelligence and Statistics*, 1618–1628. PMLR.

Mandal, D.; Procaccia, A. D.; Shah, N.; and Woodruff, D. P. 2019. Efficient and thrifty voting by any means necessary. *Advances in Neural Information Processing Systems*.

Neu, G.; and Valko, M. 2014. Online combinatorial optimization with stochastic decision sets and adversarial losses. In *Neural Information Processing Systems*.

Parkes, D. C.; and Procaccia, A. D. 2013. Dynamic Social Choice with Evolving Preferences. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 767–773.

Procaccia, A. D.; and Rosenschein, J. S. 2006. The distortion of cardinal preferences in voting. In *International Workshop on Cooperative Information Agents*, 317–331. Springer.

Roth, A. E.; Sönmez, T.; and Ünver, M. U. 2004. Kidney Exchange*. *The Quarterly Journal of Economics*, 119(2): 457–488.

Roth, A. E.; and Sotomayor, M. 1992. Two-sided matching. *Handbook of game theory with economic applications*, 1: 485–541.

Sankararaman, A.; Basu, S.; and Sankararaman, K. A. 2021. Dominate or Delete: Decentralized Competing Bandits in Serial Dictatorship. In *International Conference on Artificial Intelligence and Statistics*, 1252–1260. PMLR.

Satzger, B.; Endres, M.; and Kießling, W. 2006. A Preference-Based Recommender System. In Bauknecht, K.; Pröll, B.; and Werthner, H., eds., *E-Commerce and Web Technologies*, 31–40.

Waldfogel, J. 2008. The median voter and the median consumer: Local private goods and population composition. *Journal of urban Economics*, 63(2): 567–582.

Woeginger, G. J. 2018. The Open Shop Scheduling Problem. In *35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

A Proofs

A.1 Proof of Theorem 1

Proof. We now consider an instance of SBM with n agents and n services, where each agent has the same preferences. That is, $\succ_a = \succ_b$ for all $(a, b) \in N$. Furthermore, assume that, without loss of generality, service j is in the jth position of this preference ordering. That is, assume that $\succ_a(j) = j$ for all $j \in S$. Lastly, assume that the blocking delay on each of the n services is $\tilde{D}$ for all agents. In other words, put more formally, assume that $D_{ij} = \tilde{D}$ for all $i \in N$ and all $j \in S$.

We proceed with the proof in the following manner. Given the matching sequence M returned by a deterministic policy using the above preference profile and blocking delay matrix, we will show that there exists a set of reward profiles which induce the preference profile, and on which the matching sequence M suffers a distortion of $O(n) = O(s)$. We construct this reward profile via an inductive argument.

Firstly, observe that there must exist some agent, $i_1 \in N$, who is assigned service 1 at most $T/\tilde{D}n$ times in the matching sequence M by the pigeonhole principle. We set the reward profile of agent i_1 to $(1, 0, \ldots, 0)$. Disregarding agent i_1, observe that there must exist different agent, $i_2 \in N$, who is assigned service 1 or 2 at most $T/\tilde{D}(n-1)$ times, once again by the pigeonhole principle. We set the reward profile of agent i_2 to $(1/2, 1/2, 0, \ldots, 0)$. Disregarding both agents i_1 i_2, we can find a new agent, $i_3 \in N$, who has been assigned services 1, 2 or 3 at most $T/\tilde{D}(n-2)$ times. we set the reward profile of agent i_3 to $(1/3, 1/3, 1/3, 0 \ldots, 0)$. We proceed in this pattern for a total of n steps, until all agents are assigned reward profile. Note that the reward profiles induce the desired preference profile (assuming that a numeric tie-breaking rule is used).

Given the assigned reward profiles, it is obvious that an optimal matching sequence assigns service j to agent i_j whenever the service is available. The social welfare of this optimal matching sequence is therefore of order $O(\log(n)T/\tilde{D})$. In contrast, the matching sequence M has social welfare of order $O(\log(n)T/\tilde{D}n)$. As we can always construct such a reward profile, no matter the matching sequence M returned by a policy, this implies that the distortion of any policy is of order $O(n) = O(s)$.

□

A.2 Proof of Theorem 2

Proof. Similarly to Theorem 1, we consider an instance of SBM with n agents and n services. Additionally, assume that the blocking delays for all services is the same for all agents. That is, $D_{ij} = d$ for some $d \leq \tilde{D}$.

Before moving to the content of the proof, we first show that it is sufficient to consider only *anonymous* policies. Given a preference profile $\succ$, we let $A_{ij}(\succ) \in \{0, 1, \ldots, T\}$ denote the random variable that indicates the number of times agent i was allocated service j. We call a randomised matching algorithm anonymous if $\mathbb{E}[A_{ij}(\succ_1, \ldots, \succ_n)] = \mathbb{E}[A_{\sigma(i)j}(\succ_{\sigma(1)}, \ldots, \succ_{\sigma(n)})]$ for all permutations σ. In other words, a matching policy is anonymous if each agent is assigned each policy the same number of times in expectation, regardless of the agents' relative positions in the preference profile.

Now, suppose we are given a matching policy which has distortion at most ρ i.e. $\sum_{ij} \mu_{ij} \mathbb{E}[A_{ij}(\succ)] \geq \rho \mathrm{OPT}(\mu)$. We can consider a new matching policy that selects a permutation σ uniformly at random and then applies the same policy on the input $\succ_\sigma = (\succ_{\sigma(1)}, \ldots, \succ_{\sigma(n)})$. Then the expected social welfare of the new policy is

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

The first inequality follows because the original policy 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 policy has distortion at most ρ. This implies that for any matching policy, there is an anonymous matching policy with identical performance with respect to distortion. As a result, from now on, we restrict our consideration to anonymous matching policies without loss of generality.

Next we will show that any anonymous matching policy incurs distortion of order $\Omega(s)$ via construction of a special set of reward profiles. The reward profiles we construct are very similar to the ones constructed in the proof of Lemma 8 of (Filos-Ratsikas, Frederiksen, and Zhang 2014). For each $i \in [\sqrt{n}]$, define

$$\mu_{i,j} = \begin{cases} 1 - \sum_{j\neq i}\mu_{i,j} & \text{if } j = i \\ \frac{n-j}{10n^3 d} & \text{o.w.} \end{cases}$$

And for each $\ell \in [\sqrt{n}-1]$, define

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

The n agents are grouped into $\sqrt{n}$ groups and all agents in group i have the same preference order. Let $G_i = \{i\} \cup \{i + \ell\sqrt{n} : \ell = 1, \ldots, \sqrt{n}-1\}$. Observe that 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 service 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 service j can be allocated at most T/d times we have

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

We now bound the expected social welfare of any randomized and anonymous matching policy with the given reward profile. For any agent $i \in [\sqrt{n}]$, the maximum expected utility over the T rounds is at most $T_{ii} + \sum_{j\neq i} T_{ij}\frac{n-j}{10n^3 d} \leq T_{ii} + O\left(\frac{T}{nd}\right)$. 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 $T_{ii}O\left(\frac{1}{\sqrt{n}}\right) + \sum_{j\neq i, j\leq\sqrt{n}} T_{ij}\frac{1}{\sqrt{n}} + \sum_{j>\sqrt{n}} T_{ij}\frac{n-j}{10n^3 d} \leq O\left(\frac{1}{\sqrt{n}}\right)\sum_{j=1}^{\sqrt{n}} T_{ij} + O\left(\frac{T}{nd}\right)$. Therefore, the total utility over all the n agents is bounded by

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

where the last line follows from equation (1). On the other hand, any deterministic and non-anonymous allocation rule that always assigns service i to agent i every D rounds achieves a social 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 of the permutation σ sampled 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 (Basu et al. 2019). 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 (Basu et al. 2019). 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 own allocation, independent of RRSD. That is, agent k is tasked with solving the following integer linear programming problem (ILP):

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

The variables $x_{t,j}$ indicate whether agent k is assigned service j at time step t. Meanwhile, the constants $y_{t,j}$ indicate whether agent k cannot be assigned service j on time step t due to blocking delay constraints imposed by allocations of service j to agents 1 through $k-1$. The second set of constraints ensure that the assignments chosen by agent k do not breach the delay constraints imposed by preexisting assignments of services to 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 assignments which obeys its own blocking delay constraints.

We will proceed to develop an upper bound on this ILP through a series of relaxations. We will then compare this upper bound to an assignment which is outperformed by `RRSD` to prove an asymptotic bound on the incentive ratio, as desired.

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 assign itself a fractional mixture of services. Additionally, we replace the constants $y_{t,j}$ with variables $z_{t,j}$ 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 assign itself each service j 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):

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

It should be immediately obvious that this problem can be reformulated further, and, in fact, the individual fractional assignments per time step can be replaced with fractional assignments of agents to services for the entire time horizon. Similarly, the newly introduced auxiliary variables $z_{t,j}$ 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:

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

Additionally, we define $C_j = \{t \in [T] \ : \ y_{t,j} = 1\}$ as the set of time steps in which agent k cannot (in practice) be matched with service j 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 service j as an item with weight $D_{k,j}$ and value $\mu_{k,j}$. From this perspective, a_j is the (fractional) number of times we pack item j into a knapsack (whose capacity is T). Note that the maximum value a_j can get in the previous LP is determined by the pattern of C_j, and is also capped by $T/D_{k,j}$. Therefore, in our bounded knapsack formulation, we can replace the constraints of a_j to be $a_j \leq T/D_{k,j} - b_j$ where b_j is the number of blocks caused by C_j. Note that in general $b_j \neq |C_j|$, as it heavily depends on the pattern of the blocks. Since $a_j \geq 0$, we have that $\frac{T}{D_{k,j}} \geq b_j$. It is well known that this fractional bounded knapsack admits the optimal solution $\forall j \in S, a^*_j = \min\{T/D_{k,j} - b_j, (T - b_j - \sum_{l=1}^{j-1} a^*_l)^+\}$. Note that the solution a^*_j implicitly specifies an upper bound for the original ILP.

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

We now define for each $j \in S$, $a'_j = T_j/D_{k,j} - b_j$, and $T_j = \left(T - \sum_{l=1}^{j-1} a'_l + |A_j \cap C_j|\right)^+$. We claim that $\sum_{l=1}^{j} a^g_l \geq \sum_{l=1}^{j} a'_l$. In turn, this immediately implies that $\sum_{j=1}^{s} a^g_j \mu_{k,j} \geq \sum_{j=1}^{s} a'_j \mu_{k,j}$. In other words, we will attempt to prove that the solution a'_j specifies a lower bound on the performance of `RRSD`.

We prove the claim using induction on j. We know that $a^g_1 \geq \lceil (T - b_1) D_{1,k} \rceil$, so the base case is satisfied. By the inductive hypothesis, assume that $\sum_{l=1}^{j} a^g_l \geq \sum_{l=1}^{j} a'_l$ for all $j < j'$. We have:

$$
\begin{aligned}
&a^g_{j'} \geq \lceil (T - |col_{j'}|)/D_{k,j'} \rceil \\
&\geq \frac{1}{D_{k,j'}} \left(T - \sum_{l=1}^{j'-1} a^g_l - b_{j'} + |A_{j'} \cap C_{j'}| \right) \\
&= \frac{1}{D_{k,j'}} \left(T - \sum_{l=1}^{j'-1} a'_l - \sum_{l=1}^{j'-1} (a^g_l - a'_l) - b_{j'} + |A_{j'} \cap C_{j'}| \right) \\
&= a'_{j'} - \frac{1}{D_{k,j'}} \sum_{l=1}^{j'-1} (a^g_l - a'_l)
\end{aligned}
$$

Thus we have that

$$\sum_{l=1}^{j'} (a^g_l - a'_l) \geq (1 - 1/D_{k,j'}) \sum_{l=1}^{j'} (a^g_l - a'_l)$$

which means $\sum_{l=1}^{j'} a^g_l \geq \sum_{l=1}^{j'} a'_l$, and the inductive hypothesis holds. In what follows, we will refer to a' as the lower bound solution. Similarly, we will refer to a^* as the upper bound solution.

Comparing the bounds. Note that for any j, if $\frac{T}{D_{k,j}} = b_j$ then both the upper bound and lower bound solutions will not contain service j (as $a'_j \leq a^*_j = 0$). Therefore, without loss of generality, we assume that $\frac{T}{D_{k,j}} > b_j$. We set $D'_{k,j}$ such that $\frac{1}{D'_{k,j}} = \frac{1}{D_{k,j}} - \frac{b_j}{T}$. With induction in j we can show that $a'_j = \frac{T}{D'_{k,j}} \prod_{l=1}^{j-1} (1 - \frac{1}{D'_{k,l}})$. In addition, we can also show that $a^*_j \leq \frac{T}{D'_{k,j}} + 1$. The remainder of the proof consists of showing that the allocation a'_j performs asymptotically well compared to the optimal allocation a^* via the closed forms above, which in turn implies the desired asymptotic bound on the incentive ratio of RRSD. The proof of this fact is contained in the proof for the blocking bandits setting provided by (Basu et al. 2019), and as a result is omitted. We point the enthusiastic reader to the 'Greedy Lower Bound vs LP Upper Bound' subsection of the proof of Theorem 3.3 in (Basu et al. 2019).

□

A.4 Proof of Theorem 4

Proof. Our proof proceeds by upper bounding the distortion of RRSD by the distortion of RSD on a new set of reward profiles. The distortion of RRSD is given as

$$\rho = \sup_{\mu, D} \frac{\text{SW}(M^*(\mu, D), \mu)}{\mathbb{E}[\text{SW}(\texttt{RRSD}(\succ, D), \mu)]}$$

We first upper bound $\text{SW}(M^*(\mu, D), \mu)$ by expected welfare on a new instance with no blocking. For this new instance, there are $\tilde{D}n$ total agents and s services. The $D_{\max}n$ agents are partitioned into n groups, one for each agent in the original profile. We will write the group of agents corresponding to agent i in the original profile by G_i and the ℓ-th agent in group G_i will be denoted by i_ℓ. The new reward profile $\tilde{\mu} \in \mathbb{R}^{D_{\max}n \times s}$ is defined as follows

$$\tilde{\mu}_{i_\ell, j} = \frac{\mu_{i,j}}{D_{ij}} \quad \forall j \, \forall i_\ell \in G_i$$

In the new instance, there is no blocking i.e. all the blocking lengths are one.

Now let $\pi^* = M^*(\mu, D)$ be the optimal policy for the original instance μ with blocking. We now construct a new policy for the non-blocking setting with reward matrix $\tilde{\mu}$. The new policy $\tilde{\pi}$ works as follows. At time t, if π^* allocated service j to agent i, then we select one available agent from the group G_i (say i_ℓ) and repeatedly allocate service j to agent i_ℓ for the next $D_{i,j}$ rounds i.e. we set $\tilde{\pi}_{t'}(i_\ell) = j$ for $t' = t, t+1, \ldots, t + D_{i,j} - 1$. Notice that, since there are $\tilde{D}$ agents in group G_i, it is always possible to find such an available service i_ℓ whose allocation hasn't been determined at round t. This is because under the original policy π^*, at any time at most $\tilde{D}$ services can be simultaneously blocked as result of being assigned to agent i. Algorithm 1 describes how to construct the new policy $\tilde{\pi}$ from the old policy π^*.

Let us now compare the social welfare of policy π^* with reward instance μ, and social welfare of policy $\tilde{\pi}$ with reward instance $\tilde{\mu}$. Notice that whenever π^* allocates service j to agent i, a corresponding agent (say i_ℓ) is assigned service j exactly $D_{i,j}$ times under the new policy $\tilde{\pi}$. As the new rewards are normalized by the blocking lengths, this implies that the total reward gathered by i under π^* is the same as the total reward gathered by all the agents in G_i under the new policy $\tilde{\pi}$. Thus, summing over all the agents, we have $\text{SW}(\pi^*, \mu) = \text{SW}(\tilde{\pi}, \tilde{\mu})$.

Now observe that, under the new instance $\tilde{\mu}$, there is no blocking, so the optimal allocation rule is obtained by applying a fixed matching (say σ^*)[4] repeatedly over the T rounds. Let $\text{SW}_0(\sigma^*, \tilde{\mu})$ be the one-round social welfare of the matching σ^*. Then under the non-blocking reward instance $\tilde{\mu}$, the best possible social welfare is $T \cdot \text{SW}_0(\sigma^*, \tilde{\mu})$. This gives us the following bound on the welfare of the original policy π^* for the blocking instance.

$$\text{SW}(\pi^*, \mu) \leq \text{SW}(\tilde{\pi}, \tilde{\mu}) \leq T \cdot \text{SW}_0(\sigma^*, \tilde{\mu}) \qquad (2)$$

We now prove a lower bound on the expected social welfare of RRSD under the original reward instance μ. RRSD is a randomized policy, but for a given order of agents, the sequence of assignments generated by RRSD becomes a deterministic policy. Any such deterministic policy π can be converted to an equivalent policy (say $g(\pi)$) through Algorithm 1. Moreover, the new policy $g(\pi)$ preserves the social welfare under the new reward instance $\tilde{\mu}$. The expected social welfare of RRSD is given as

$$
\begin{aligned}
\mathbb{E}[\text{SW}(\texttt{RRSD}, \mu)] &= \mathbb{E}_{\pi \sim \texttt{RRSD}}[\text{SW}(\pi, \mu)] \\
&= \mathbb{E}_{\pi \sim \texttt{RRSD}}[\text{SW}(g(\pi), \tilde{\mu})]
\end{aligned}
$$

[4] Ideally σ^* is an assignment from $D_{\max}n$ agents to s services based on the optimal achievable social welfare and not a one-to-one matching. But we will use the term matching instead of assignment to be consistent with the rest of the paper.

Algorithm 1: Policy Conversion ($\pi^* \to \tilde{\pi}$).

```
Input: $T, N, S, D$, Policy $\pi^*$
Output: $\tilde{\pi}$
/* Matrix $F$ keeps track of the
   available agents within each
   group $G_i$.                              */
$F(i, i_\ell) = 0 \; \forall i_\ell \in G_i \; \forall i \in N$
for $t \in \{1, \dots, T\}$ do
    for $i \in N$ do
        if $\pi^*_t(i) = j$ then
            /* Find an available agent
               within group $G_i$              */
            Choose $i_\ell$ s.t. $F(i, i_\ell) = 0$
            $\tilde{\pi}_{t'}(i_\ell) = j \; \forall t' \in [t, \dots, t + D_{ij} - 1]$
            $F(i, i_\ell) = 1$
        end
        /* If any arm becomes
           available under $\pi^*$, then we
           make corresponding agents
           available                           */
        for $j \in S$ do
            if $\pi^*_{t-D_{i,j}+1}(i) = j$ then
                /* $i_j \in G_i$ is the
                   corresponding agent
                   with repeated
                   allocations from
                   $\{t - D_{i,j} + 1, \dots, t\}$ under $\tilde{\pi}$
                   */
                $F(i, i_j) = 0$
            end
        end
    end
end
return $\tilde{\pi}$
```

The last line follows from the welfare preservation property of Algorithm 1. Moreover, given a policy π, the new policy $g(\pi)$ actually assigns the same service repeatedly to the same agent. This is because whenever the original RRSD assigns a new service to agent i, algorithm 1 selects a new agent in G_i, and assigns the new service to the new agent. Now given $g(\pi)$ consider a simpler policy $g'(\pi)$ which only makes the first repeated assignments to any member from G_i for all i. Since this new policy makes fewer assignments than $g(\pi)$ we have the following inequality.

$$\mathbb{E}_{\pi \sim \texttt{RRSD}}[\text{SW}(g(\pi), \tilde{\mu})] \geq \mathbb{E}_{\pi \sim \texttt{RRSD}}[\text{SW}(g'(\pi), \tilde{\mu})]$$

Since the sequence π was generated under RRSD for the original instance μ, an alternative way to generate the sequence $g'(\pi)$ is the following: first random choose an order of the set N, and then replace each agent i in this sequence with a randomly choose agent from G_i. Let us call this new randomised policy GRSD (short for grouped RSD). Then we have,

$$\begin{aligned}\mathbb{E}_{\pi \sim \texttt{RRSD}}[\text{SW}(g'(\pi), \tilde{\mu})] &= T \cdot \mathbb{E}[\text{SW}_0(\texttt{GRSD}, \tilde{\mu})] \\ &= T \cdot \mathbb{E}[\text{SW}_0(\texttt{RSD}, \tilde{\mu})]\end{aligned}$$

The last equality follows from the following two observations. Under GRSD, the probability that an agent from group i shows up at position j equals $1/n$. On the other hand, under RSD, the probability that an agent from group i shows up at position j equals $\tilde{D} \times (1/\tilde{D}n) = 1/n$. Second, conditioned on the event an agent from group i shows up at position j, the expected utility of the agent is the same, as all the agents in group G_i are duplicates of the original agent i and have the same reward profile. Therefore, we have established the following lower bound on the expected welfare of RRSD under the original instance μ.

$$\mathbb{E}[\text{SW}(\texttt{RRSD}, \mu)] \geq T \cdot \mathbb{E}[\text{SW}_0(\texttt{RSD}, \tilde{\mu})] \tag{3}$$

We can now bound the distortion of RRSD as follows.

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

Where the inequality is due to the lower bound from eq. 3 and the upper bound from eq. 2. Since the last quantity is just the distortion of RSD in the one-shot matching setting, we can apply Lemma 4 from (Filos-Ratsikas, Frederiksen, and Zhang 2014) and get a bound of $O(\sqrt{s})$ on the distortion.[5] □

A.5 Derandomised RRSD

In this section, we present a deterministic matching policy for the offline SBM setting. More precisely, we present derandomised RRSD (DRRSD), which, as the name suggests, is a derandomised version of RRSD. Instead of sampling a single permutation, like RRSD, DRRSD uses a set of $4n^2 \log(n)$ permutations. In addition, this set is constrained to ensure that the fraction of permutations in which an agent i appears in the jth position is at least $\frac{1}{2n}$. The following lemma stipulates that such a set of permutations always exists.

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

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

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

Moreover, by a union bound over the n agents and n positions we get that

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

[5] Filos-Ratsikas, Frederiksen, and Zhang (2014) actually considered a setting where $n = s$, but their proof naturally generalizes for the setting with $n > s$

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. □

DRRSD splits the time horizon into evenly sized blocks. In each block, a different permutation is used. Within each block, agents are assigned to services by the same greedy method used by RRSD, with one caveat. If the blocking delay caused by the assignment of an agent-service pair would overrun into the next block, then this assignment is skipped. This to ensure that all services will be available at the beginning of each block. The pseudocode for DRRSD is presented in Algorithm 2.

Algorithm 2: DRRSD (Derandomized RRSD)

```
Input: $T$, $N$, $D$, $S$, $\succ$, and a set of $P = 4n^2\log(n)$
       permutations $\{\sigma_1, \ldots, \sigma_P\}$
$M = (m_t)_{t=1}^T = (\emptyset)_{t=1}^T$
for $p = 1, \ldots, P$ do
    $\sigma = \sigma_p$
    for $i = 1, \ldots, n$ do
        // Select agent
        ag $= \sigma(i)$
        start $= (p-1)T/P$
        end $= pT/P$
        for $j = 1, \ldots, s$ do
            // Select service
            ser $= \tilde{\succ}_{\text{ag}}(j)$
            while $available(M, ag, ser, start, end)$ do
                $t =$ earliest($M$, ag, ser, start, end)
                if $overrun(ag, ser, t, end)$ then
                    break
                end
                $M(t, \text{ag}) =$ ser
            end
        end
    end
end
return $M$
```

Next, we prove that DRRSD incurs a distortion of order $O(s)$, which matches the lower bound we established for the distortion of deterministic policies in Theorem 1.

A.6 Proof of Theorem 5

Proof. We will write i_j to denote agent i's jth favourite service. That is, $i_j = \succ_i(j)$. By Lemma 1, agent i gets her jth favourite service (or better) in at least $\frac{P}{2n}$ groups. Within any such group, there are T/P time slots, and agent i is assigned her jth favourite service at least $\lfloor T/(PD_{i,i_j}) \rfloor$ times. Therefore, the total welfare guaranteed by DRRSD is at least

$$\begin{aligned} &\sum_{i=1}^{n}\sum_{j=1}^{S} \mu_{i,i_j} \frac{P}{2n} \left\lfloor \frac{T}{PD_{i,i_j}} \right\rfloor \\ \geq &\sum_{i=1}^{n}\sum_{j=1}^{s} \mu_{i,i_j} \frac{P}{4n} \frac{T}{PD_{i,i_j}} \\ = &\frac{T}{4n} \sum_{i=1}^{n}\sum_{j=1}^{s} \frac{\mu_{i,i_j}}{D_{i,i_j}} \end{aligned}$$

On the other hand, consider a matching algorithm that assigns service 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 $4n = O(s)$. □

A.7 Proof of Theorem 6

The pseudocode for BRRSD is given in Algorithm 3. We now restate the three claims of the Theorem 6, and proceed with a proof below:

(i) The dynamic $(1/\sqrt{s})$-regret of BRRSD is $O\left(\tilde{D}\sqrt{s}\log\left(Tsn\right)/\Delta^2_{min}\right)$.

(ii) The $(1-1/e)$-IC regret for all agents under BRRSD is $O\left(\tilde{D}s\log\left(Tsn\right)/\Delta^2_{min}\right)$.

(iii) The greedy algorithm used by BRRSD in the exploration phase uses at most twice as many time steps as the shortest feasible matching sequence which completes the required assignments.

Proof. Claim (i) can be proved as follows. By the end of the exploration phase, we know that each agent has received a reward from being assigned each service at least $\lceil 2\log(2Tsn)/\Delta^2_{\text{min}} \rceil$ times. In addition, note that this exploration phase takes at most $\tilde{D}s\lceil 2\log(2Tsn)/\Delta^2_{\text{min}} \rceil + \tilde{D}$ rounds. By the Chernoff-Hoeffding inequality, we have that for all agents i and services j:

$$P\left(|\mu_{i,j} - \hat{\mu}_{i,j}| \geq \frac{\Delta_{\text{min}}}{2}\right) \leq \frac{1}{Tsn}$$

Thus, by the union bound and the assumption that all agents are mean-based, with probability $1-1/T$, the internal estimation of every agent will be correct. From now, unless explicitly stated, we will assume that all agents have learned the correct preference ordering by the end of the exploration phase.

For the sake of simplicity, let T_1 denote the number of rounds for which the exploration phase runs, and let T_2 denote the number of rounds for which the exploitation phase runs. Similarly let OPT_1 denote the social welfare of the optimal matching sequence of length T_1, and OPT_2 denote the social welfare of the optimal matching sequence of length T_2. Furthermore, let SW_1(BRRSD) denote the social welfare generated by BRRSD in the exploration phase, and SW_2(BRRSD) denote the social welfare generated by BRSSD in the exploitation phase.

As each reward is bounded between $[0,1]$, and the exploration phase proceeds for at most $\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\text{min}}^2\right\rceil+\tilde{D}$ time steps, it is easy to show that

$$\frac{1}{\sqrt{s}}\mathbb{E}\left[\text{OPT}_1\right]-\mathbb{E}\left[\text{SW}_1(\texttt{BRRSD})\right]\leq$$
$$\frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\text{min}}^2\right\rceil+\tilde{D}\right)$$

In addition, by Theorem 4, and the assumption that agents are mean-based, we have the following lower bound:

$$\frac{1}{\sqrt{s}}\mathbb{E}\left[\text{OPT}_2\right]\leq\mathbb{E}\left[\text{SW}_2(\texttt{BRRSD})\right]$$

Let OPT denote the social welfare of the optimal matching sequence of length T. Combining the bounds above and noting that $\text{OPT}\leq\mathbb{E}[\text{OPT}_1]+\mathbb{E}[\text{OPT}_2]$ we have:

$$\frac{1}{\sqrt{s}}\mathbb{E}[\text{OPT}]-\mathbb{E}[\text{SW}(\texttt{BRRSD})]\leq$$
$$\frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\text{min}}^2\right\rceil+\tilde{D}\right)$$

For the case when at least 1 agent does not learn the correct preference ordering by the end of the exploration phase, the $(\frac{1}{\sqrt{s}})$-dynamic regret of BRRSD is bounded above by $\frac{1}{\sqrt{s}}nT$. Combining both cases together, we see that the dynamic $(\frac{1}{\sqrt{s}})$-regret of BRRSD is bounded above by $\frac{1}{\sqrt{s}}\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\text{min}}^2\right\rceil+\tilde{D}\right)+\frac{1}{\sqrt{s}}n$, implying the desired regret bound.

To prove claim (ii), note that the an agent can only affect its assignment of services in the exploitation phase, in which BRRSD deploys the RRSD algorithm. Thus, following a similar argument as above, replacing the use of Theorem 4 with Theorem 3, we find that the $(1-1/e)$-IC regret of BRRSD is bounded above by $(1-1/e)\left(\tilde{D}s\left\lceil 2\log(2Tsn)/\Delta_{\text{min}}^2\right\rceil+\tilde{D}\right)+(1-1/e)n$.

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 j is a set of n independent tasks $j_1,\ldots,j_n$. The task j for job i must be processed on machine i for an uninterrupted $D_{i,j}$ time units. A schedule assigns every task j_i to a time interval $D_{i,j}$ 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 the exploration phase of BRRSD reduces to an open shop scheduling problem in which there is a job for each agent i, and a task for each assignment of service j to agent i. Similarly, observe that the assignment procedure used by BRRSD is simply an implementation of the greedy algorithm for open shop scheduling as described by (Woeginger 2018). The claim follows from that fact that the greedy algorithm is a 2-approximation for open shop scheduling (see (Woeginger 2018)).

□

B Computational complexity of Offline SBM

In this section, we investigate the computational complexity of the offline SBM setting. Observe that, when there is only one agent, offline SBM corresponds to the stochastic blocking bandit problem defined in (Basu et al. 2019). Therefore, offline SBM inherits the complexity issues of the offline stochastic blocking bandit problem. More precisely, there is no pseudopolynomial time algorithm for offline SBM setting unless the randomised exponential time hypothesis (Calabro et al. 2008) is false.

Theorem 7. *Offline SBM problem does not admit a pseudopolynomial time policy unless the randomised exponential time hypothesis is false.*

C Additional Pseudocode

In this section, for clarity, we provide additional pseudocode regarding the subroutines used by RRSD, DRRSD, and BRRSD. Pseudocode for available(), which checks whether a service can be assigned to an agent, is given in Algorithm 4. The pseudocode for earliest(), which checks the earliest time step in which an agent can be assigned a service, is given in Algorithm 5. The pseudocode for overrun(), which checks whether an assignment of a service in one block of DRRSD will cause a blocking delay in the next block, is given in Algorithm 6. Lastly, the pseudocode for buildJobList(), which specifies the agent-service assignments for BRRSD to complete during the exploration phase, is given in Algorithm 7.

Algorithm 3: BRRSD (Bandit RRSD)

```
Input: $T$, $N$, $D$, $S$, $\Delta_{\text{min}}$
$M = (m_t)_{t=1}^{T} = (\emptyset)_{t=1}^{T}$
repeats $= \lceil 2\log(2Tsn)/\Delta_{\text{min}}^2 \rceil$
// Build a list of exploration assignments
jobList $=$ buildJobList$(n, s, \text{repeats})$
explore $=$ true
waiting $=$ false
count $= 0$
for $t = 1, \ldots T$ do
    // Exploration phase
    if explore then
        // Greedily add remaining agent-service pairs to current matching
        for $(i, j)$ in jobList do
            if available($M$, $i$, $j$, $t$) then
                $M(t, i) = j$
                jobList.remove$(i, j)$
            end
        end
        // Start waiting phase
        if jobList.isEmpty then
            explore $=$ false
            waiting $=$ true
        end
    end
    // Wait for all services to become available
    if waiting then
        if count $< \tilde{D}$ then
            count++
        end
        // Start exploitation phase
        else
            $\tilde{\succ} = \tilde{\succ}^t$
            Sample $\sigma$
            waiting $=$ false
        end
    end
    // Exploitation phase
    else
        for $i = 1, \ldots n$ do
            ag $= \sigma(i)$
            for $j = 1, \ldots, s$ do
                ser $= \tilde{\succ}_{\text{ag}}(j)$
                if available$(M, ag, ser, t)$ then
                    $M(t, \text{ag}) = \text{ser}$
                end
            end
        end
    end
end
return $M$
```

Algorithm 4: Different versions of the auxiliary process available

```
Function available($M$, $i_1$, $j$, $t_0$):
    if $M(t_0, i_1) \neq 0$ then
        return false
    end
    for $i = 1, \ldots, n$ do
        for $t = t_0 - D_{i,j} + 1, \ldots, t_0 + D_{i,j} - 1$ do
            if $M(t, i) = j$ then
                return false
            end
        end
    end
    return true
End Function
Function available($M$, $i$, $j$, start, end):
    for $t =$ start$, \ldots,$ end do
        if available$(M, i, j, t)$ then
            return true
        end
    end
    return false
End Function
Function available($M$, $i$, $j$):
    start $= 1$
    end $= T$
    return available($M$, $i$, $j$, start, end)
End Function
```

Algorithm 5: Different versions of the auxiliary process earliest

```
Function earliest($M$, $i$, $j$, start, end):
    for $t =$ start$, \ldots,$ end do
        if available$(M, i, j, t)$ then
            return $t$
        end
    end
    // Failure state
    return 0
End Function
Function earliest($M$, $i$, $j$):
    start $= 1$
    end $= T$
    return earliest$(M, i, j, start, end)$
End Function
```

Algorithm 6: The auxiliary process overrun

```
Function overrun($M$, $i$, $j$, $t$, end):
    if $t + D_{i,j} - 1 \geq$ end then
        return false
    end
    return true
End Function
```

Algorithm 7: The auxiliary process buildJobList

```
Function buildJobList(n, s, repeat):
    list = []
    for $i = 1, \ldots, n$ do
        for $j = 1, \ldots, s$ do
            for $k = 1, \ldots,$ repeat do
                list.append(($i$, $j$))
            end
        end
    end
End Function
```