Learning Equilibria in Matching Markets from Bandit Feedback

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large-scale two-sided matching platforms must find market outcomes that align with user preferences while simultaneously learning these preferences from data. However, since preferences are inherently uncertain during learning, the classical notion of stability (Gale and Shapley, 1962; Shapley and Shubik, 1971) is unattainable in this setting. To bridge this gap, we develop a framework and algorithms for learning stable market outcomes under uncertainty. Our primary setting is matching with transferable utilities, where the platform both selects a matching and sets monetary transfers between agents. We introduce an economically motivated measure of instability, which we define to be the minimum amount the platform could subsidize agents to achieve stability. Using this measure as a loss function, we design low-regret algorithms for learning stable matchings from noisy user feedback in a multi-armed bandits model. Our core algorithmic insight is that "optimism in the face of uncertainty," the principle underlying many bandit algorithms, applies to a primal-dual formulation of matching with transfers and leads to near-optimal regret bounds. Finally, we show that our approach extends to structured preferences and to matching with non-transferable utilities.

1 Introduction

Data-driven marketplaces face the simultaneous challenges of learning agent preferences and aligning market outcomes with agent incentives. Consider, for instance, online platforms that match market participants from two sides to each other (e.g., Lyft, TaskRabbit, and Airbnb). If participants on either side are not offered desirable matchings at fair prices, they would have incentive to leave the platform and switch to a competing platform. User preferences, however, are often unknown to the platform and must be learned. This is the focus of the present paper: When faced with uncertainty about user preferences (and thus incentives), *how should a marketplace explore and learn market outcomes that align with user incentives?*

We investigate this question under a model called *matching with transferable utilities*, proposed by Shapley and Shubik [30]. In this model, there is a two-sided market of heterogeneous agents (e.g., customers and service providers). Each customer has a utility they derive from being matched to a given provider, and vice versa. The platform selects a matching between the the two sides and assigns a monetary transfer between each pair of matched agents. An agent's net utility is their value for being matched to their partner plus the value of their transfer (either of which can be negative in the cases of costs and payments). Transfers are a salient feature of most real-world matching markets: riders pay drivers on Lyft, clients pay freelancers on TaskRabbit, and guests pay hosts on Airbnb.

When preferences are known, the concept of stability formalizes compatibility with user preferences, delineating market outcomes where no user has incentive to match outside the platform. A matching with transfers is *stable* if no pair of agents can agree on a transfer such that both would rather match

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

with each other than abide by the matching and transfers. Shapley and Shubik [30] show that stable outcomes are "socially optimal," in that they maximize the sum of agent utilities. In fact, any socially optimal matching can be made stable with appropriate transfers. They also show stable outcomes coincide with market equilibria, when viewing transfers as placing an implicit price on each agent.

However, in the context of large-scale matching platforms, the assumption that preferences are known breaks down. Platforms usually cannot have users report their complete preference profiles. Moreover, users may not even be aware of what their own preferences are. For example, a freelancer may not exactly know what types of projects they prefer until actually trying out specific ones. In reality, the platform is more likely to learn information about preferences from repeated feedback[1] over time.

Given that preferences are inherently uncertain while learning, the standard Shapley-Shubik setup demands too much information. Ensuring (or even verifying) that a matching with transfers is stable requires detailed knowledge of user preferences. Consequently, with uncertainty, the platform cannot hope to achieve stability in the traditional sense. To address this issue, we lean on approximation, focusing on learning market outcomes that are *approximately* stable. Two questions now arise: How should one quantify whether a matching with transfers is "far from stable"? And given such a measure, how should a market platform learn approximately stable matchings with transfers?

Our Contribution. We introduce an economically meaningful notion of *instability* for the Shapley-Shubik setting. Specifically, we measure how much the platform would have to subsidize participants to keep them on the platform and make the resulting matching stable. Our instability measure, which we call Subsidy Instability, has an equivalent dual interpretation as the maximum "unhappiness" of any subset of agents, defined as the gain in utility they could have derived from an alternate matching. Subsidy Instability also satisfies a number of properties that are important for learning: it is zero if and only if the matching is stable; it is Lipschitz continuous with respect to the agents' utility values; and it is always at least as large as the utility gap relative to the socially optimal matching.

Using Subsidy Instability, we investigate the problem of learning from noisy user feedback, which we formalize in a multi-armed bandit model. In each round, the platform selects a matching with transfers, with the goal of minimizing cumulative instability. To achieve sublinear regret, not only must the selected matchings converge to the socially optimal matching, but the selected transfers must also asymptotically guarantee stability for that matching. Our main algorithmic contribution is an optimal (up to logarithmic factors) regret bound for this problem setting:

Theorem 1.1 (Informal; see Theorem 4.3)**.** *There is an algorithm that incurs $\widetilde{O}(N^{3/2}T^{1/2})$ regret according to Subsidy Instability after T rounds, where N is the number of agents on the platform.*

Our algorithm also achieves $O(\log T)$ instance-dependent regret, where the hidden constant depends polynomially on both N and an instance-dependent utility gap (Theorem D.1). In Lemma 4.4, we show that our algorithm is optimal (up to logarithmic factors) by giving a nearly tight lower bound for the easier problem of learning a socially optimal (i.e., maximum weight) matching.

Our algorithm is based on a primal-dual formulation of matching with transfers, in which the dual variables can be used to set the transfers [30]. Our key insight is that "optimism in the face of uncertainty," which underlies many UCB-style bandit algorithms [3, 21], can be adapted to our setting. The algorithm is quite simple: we maintain upper confidence bounds on the agent utilities and in each round use these to select a maximum matching and corresponding transfers.

To analyze this algorithm, we use two ideas. First, we leverage the fact that any dual-feasible solution for the upper confidence bounds on agent utilities is also dual feasible for the true utilities. Next, we show that Subsidy Instability can be upper bounded by the difference between this dual feasible solution and a (potentially infeasible) dual solution that corresponds to the true utilities. Together, these bound the instability by the difference between the upper confidence bound and true utilities, which we analyze via a standard bandits argument.

An appealing aspect of our algorithmic approach is that it composes cleanly with analyses of existing UCB-style algorithms. To illustrate this, we extend our results to structured utility functions and to matching with non-transferable utilities (Section 4.4). Overall, our results demonstrate that incentive-aware learning in matching markets can often be *composed* onto incentive-free approaches in a simple manner, without any loss in performance.

[1] Feedback might arise from explicit sources (e.g., riders rating drivers after a Lyft ride) or implicit sources (e.g., engagement metrics on an app); in either case, feedback is likely to be sparse and noisy.

1.1 Related work

Starting with Das and Kamenica [11] and Liu et al. [22], several works [11, 22, 28, 23, 6, 5] study learning stable matchings from bandit feedback in the Gale-Shapley stable marriage model [16]. A major difference between this setting and ours is the absence of monetary transfers between agents. These works focus on the *utility difference* rather than the instability measure that we consider. Cen and Shah [6] extend this bandits model to incorporate restricted forms of transfers. However, they do not learn the transfers, and they also consider a weaker notion of stability that does not account for pairs of agents agreeing on alternate transfers.

Several papers also consider the complexity of finding stable matchings in other feedback and cost models, e.g., communication complexity [17, 2, 31] and query complexity [12, 2]. Of these works, Shi [31], which studies the communication complexity of finding approximately stable matchings with transferable utilities, is perhaps most similar to ours. This work assumes agents know their preferences and focuses on the communication bottleneck, whereas we study the costs associated to learning preferences. Moreover, the approximate stability notion in Shi [31] is the maximum unhappiness of any *pair* of agents, whereas Subsidy Instability is equivalent to the maximum unhappiness over any *coalition* of agents. For learning stable matchings, Subsidy Instability has the advantages of being more fine-grained and having a primal view that motivates a clean UCB-based algorithm.

Multi-armed bandits have also been applied to learning in other economic contexts. For example, learning a socially optimal matching (without learning transfers) is a standard application of combinatorial bandits [7, 15, 8, 10, 20]. Other applications at the interface of bandit methodology and economics include dynamic pricing [26, 18, 4], incentivizing exploration [14, 24], and learning under competition [1].

2 Preliminaries

The foundation of our framework is the matching-with-transfers model of Shapley and Shubik [29]. We begin by giving a brief account of this model and defining stable matchings with transfers.

Matching with transferable utilities. We consider a two-sided market that consists of a finite set $\mathcal{I}$ of customers on one side and a finite set $\mathcal{J}$ of providers on the other. Let $\mathcal{A} := \mathcal{I} \cup \mathcal{J}$ be the set of all agents. A *matching* $X \subseteq \mathcal{I} \times \mathcal{J}$ is a set of pairs (i, j) that are pairwise disjoint, representing the pairs of agents that are matched. For notational convenience, we define for each matching X its equivalent functional representation $\mu_X : \mathcal{A} \to \mathcal{A}$, where $\mu_X(i) = j$ and $\mu_X(j) = i$ for all pairs $(i, j) \in X$, and $\mu_X(a) = a$ if $a \in \mathcal{A}$ is unmatched.

In addition to choosing a matching, the platform chooses a vector $\tau \in \mathbb{R}^{|\mathcal{A}|}$ of transfers, where τ_a is the amount of money transferred from the platform to agent a for each $a \in \mathcal{A}$. Transfers are typically required to be *zero-sum*: $\tau_i + \tau_j = 0$ for all $(i, j) \in X$. If the transfer is a payment from customer to provider, it might be positive for the provider (who receives the payment) and negative for the customer (who makes the payment), but this is not required in the model. In Section 3, we will later relax the requirement that the transfers are zero-sum by augmenting the transfers with subsidies.

When a pair of agents $(i, j) \in \mathcal{I} \times \mathcal{J}$ matches, each experiences a utility gain: we let $u_i(j)$ and $u_j(i)$ denote the utility gains experienced by agents i and j, respectively. We allow $u_i(j)$ and $u_j(i)$ to be negative, if matching results in a net cost (e.g., if one is providing a service). We assume that each agent $a \in \mathcal{A}$ receives zero utility if unmatched: $u_a(a) = 0$. The net utility that an agent a derives from a matching with transfers (X, τ) is therefore $u_a(\mu_X(a)) + \tau_a$.

Stable matchings. In matching theory, stability captures when a matching outcome aligns with individual agents' preferences. Roughly speaking, a matching with transfers (X, τ) is stable if (i) no individual agent a would rather be unmatched, and (ii) no pair of agents (i, j) can agree on a transfer such that both would prefer matching with each over matching according to (X, τ). Formally:

Definition 2.1. A matching with zero-sum transfers (X, τ) is *stable* if it is (i) *individually rational*, i.e., $u_a(\mu_X(a)) + \tau_a \geq 0$ for all $a \in \mathcal{A}$, and (ii) *has no blocking pairs*, i.e.,

$$\big(u_i(\mu_X(i)) + \tau_i\big) + \big(u_j(\mu_X(j)) + \tau_j\big) \geq u_i(j) + u_j(i). \tag{1}$$

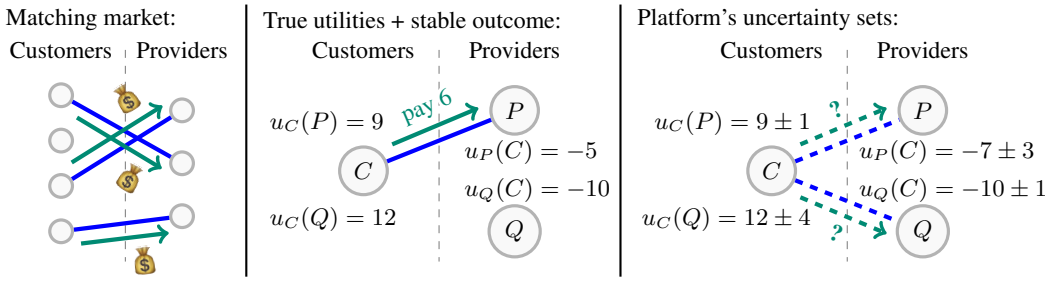

Figure 1: The left panel depicts a schematic of a matching (blue) with transfers (green). The center panel depicts a matching market with three agents and a stable matching with transfers for that market. (If the transfer 6 is replaced with any value between 5 and 7, the outcome remains stable.) The right panel depicts the same market, but with utilities replaced by uncertainty sets; note that no matching with transfers is stable for all realizations of utilities.

for all pairs $(i, j) \in \mathcal{I} \times \mathcal{J}$.[2]

A fundamental property of matchings with transfers is that if (X, τ) is stable, then X is a maximum weight matching: X maximizes the sum $\sum_{a \in \mathcal{A}} u_a(\mu_X(a))$ [30]. Shapley and Shubik [30] further show that stability coincides with Walrasian equilibria (a standard notion of equilibrium in economics). To be self-contained, we prove these classical facts in Appendix A.

To make the dynamics of the Shapley-Shubik model concrete, we provide as an example a simple market involving three agents. We will use this simple market, depicted in the center panel Figure 1, as a running example throughout the paper. The market consists of a single customer Charlene and two providers Percy and Quinn, which we record as $\mathcal{I} = \{C\}$ and $\mathcal{J} = \{P, Q\}$. If the agents' utilities are as given in Figure 1, then Charlene would prefer Quinn, but Quinn's cost of providing the service is much higher; this makes matching Charlene and Percy necessary for a stable outcome. Note that this matching is stable for any transfer from Charlene to Percy in the interval $[5, 7]$.

3 Subsidy Instability

When learning stable matchings, we must settle for guarantees of approximate stability, since exact stability—a binary notion—is impossible to achieve when preferences are uncertain. To see this, we return to the example from Figure 1. Suppose that the platform has uncertainty sets given by the right panel. Recall that for the true utilities, all stable outcomes match Charlene with Percy. If the true utilities were instead the upper bounds of each uncertainty set, then all stable outcomes would match Charlene and Quinn. Given only the uncertainty sets, it is impossible for the platform to find an (exactly) stable matching, so it is necessary to introduce a measure of approximate stability as a relaxed benchmark for the platform; we turn to this next.

3.1 Defining an instability measure

Given the insights of Shapley and Shubik [30]—that stable outcomes maximize social welfare—it is tempting to measure distance from stability simply in terms of the *utility difference*

$$\max_{X'} u_a(\mu_{X'}(a)) - \sum_{a \in \mathcal{A}} u_a(\mu_X(a)). \tag{2}$$

However, this formulation ignores transfers entirely. In fact, the utility difference can be zero even if the transfers make the matching with transfers far from stable. (See Appendix B for an example.) This reflects the fact that utility difference is *not* incentive-aware, making it unsuitable as an objective for learning stable matchings with transfers.

[2] To see that the no blocking pairs property coincides with (ii) in the informal definition above, notice that a pair (i, j) violates (1) if and only if they can find a transfer $\tau_i' = -\tau_j'$ such that $u_i(j) + \tau_i' > u_i(\mu_X(i)) + \tau_i$ and $u_j(i) + \tau_j' > u_j(\mu_X(j)) + \tau_j$.

We instead propose an economically meaningful measure of instability: *the minimum amount the platform could subsidize agents so that the subsidized matching with transfers is individually rational and has no blocking pairs*. Recalling Definition 2.1, our instability measure is defined as follows:

Definition 3.1 (Subsidy Instability)**.** Given utilities u, the *Subsidy Instability* $I(X, \tau; u)$ of a matching with transfers (X, τ) is

$$\begin{aligned} \min_{s \in \mathbb{R}^{|\mathcal{A}|}} \sum_{a \in \mathcal{A}} s_a & & (\dagger) \\ \text{s.t. } \big(u_i(\mu_X(i)) + \tau_i + s_i\big) + \big(u_j(\mu_X(j)) + \tau_j + s_j\big) \geq u_i(j) + u_j(i) & & \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\ u_a(\mu_X(a)) + \tau_a + s_a \geq 0 & & \forall a \in \mathcal{A} \\ s_a \geq 0 & & \forall a \in \mathcal{A}. \end{aligned}$$

The first set of constraints ensures that there are no blocking pairs, while the second set of constraints ensures that individual rationality is satisfied. The final $s_a \geq 0$ constraint is necessary to ensure that the matching is incentive-aware (without it, (†) would reduce to the utility difference).

Equipped with the definition of Subsidy Instability, we revisit the example in Figure 1. Consider the matching $X = \{(C, Q)\}$ with transfers $\tau_C = -11$ and $\tau_Q = 11$. (This is stable for the upper bounds of the uncertainty sets of the platform in Figure 1, but not stable for the true utilities.) The Subsidy Instability of (X, τ) is $I(X, \tau; u) = 3$ because we would need to subsidize C and P a total of at least 3. In contrast, the utility difference is 2.

3.2 Economic interpretations of Subsidy Instability

To justify our measure of instability, we provide two economic interpretations of Subsidy Instability: (i) as the platform's *cost of learning* and (ii) as a measure of *user unhappiness*.

Subsidy Instability as the platform's cost of learning. We first interpret Subsidy Instability as a pessimistic bound on the platform's *cost of learning*—how much the platform would have to pay for all users to stay on the platform in the presence of a perfect (but budget-balanced) competitor. To see this, we rewrite (†) as the minimum amount in subsidies that the platform could pay so that the subsidized matching is stable, where we extend the definition of stability (Definition 2.1) to transfers that need not be zero-sum:

$$I(X, \tau; u) = \inf_{s \in \mathbb{R}^{|\mathcal{A}|}_{\geq 0}} \left\{ \sum_{a \in \mathcal{A}} s_a : (X, \tau + s) \text{ is stable} \right\}.$$

Later on, we will see that our algorithmic approach can be extended to efficiently compute feasible subsidies for (†) such that the subsidies are within a constant factor of our regret bound. The total amount paid by the platform in subsidies thus provides a concrete realization of this "cost of learning."

Subsidy Instability as user unhappiness. The dual of the linear programming formulation (†) of Subsidy Instability lets us interpret instability from the agents' perspective rather than the platform's. At a high level, the dual program computes the *maximum unhappiness* of any subset of agents, where unhappiness is measured relative to any alternate matching with transfers (e.g., one offered by a competing platform). Formally, the *unhappiness* of a subset of agents (a "coalition") $\mathcal{C} \subseteq \mathcal{A}$ with (X, τ) is the maximum gain in total utility relative to (X, τ) that the members of $\mathcal{C}$ could achieve by matching only among themselves, such that no member is worse off than they were in (X, τ). We show the following equivalence:

Proposition 3.2 (Informal)**.** *The instability $I(X, \tau; u)$ is the maximum unhappiness of any coalition $\mathcal{C} \subseteq \mathcal{A}$ with respect to (X, τ).*

This perspective recovers the strong ε-core from coalitional game theory [29]: a matching (X, τ) belongs to the *strong ε-core* if no coalition of agents has unhappiness more than ε. Thus, $I(X, \tau; u)$ is the smallest ε such that (X, τ) belongs to the strong ε-core. This shows that Subsidy Instability has a natural interpretation in the broader context of coalitional game theory.

3.3 Learning stable matchings with bandit feedback

Using Subsidy Instability, we instantiate the platform's learning problem in a stochastic bandits framework. We assume the platform has access to *semi-bandit* feedback, with the platform receiving the utilities of every matched pair (i, j) at each round. Formally, in the t-th round:

1. A set $\mathcal{I}$ of customers and a set $\mathcal{J}$ of providers arrive to the market.
2. The platform selects a matching with *zero-sum* transfers (X^t, τ^t) between $\mathcal{I}$ and $\mathcal{J}$.
3. The platform observes noisy utilities $u_a(\mu_X(a)) + \varepsilon_{a,t}$ for each $a \in \mathcal{A}$.
4. The platform incurs as loss the instability $I(X^t, \tau^t; u)$ of the chosen matching with transfers.

In particular, the platform's "regret" is the cumulative instability: $R_T = \sum_{t=1}^{T} I(X^t, \tau^t; u)$.

In the simplest setting, the users that arrive are the same from round to round, and each agent's utility function u_a takes on values in $[-1, 1]$. Our framework extends to settings where different agents arrive in different rounds and when utility functions are structured (e.g., where each agent's utility function can be represented by a linear function of an unknown feature vector). For simplicity of exposition, we focus primarily on analyzing this setting; in Section 4.4, we extend our results to more general preference structures.

While our framework bears some resemblance to the combinatorial bandits problem of learning a maximum weight matching, there are two crucial differences that differentiate our setting: (i) in each round, the platform must choose *transfers* in addition to a matching, and (ii) loss is measured with respect to *instability* rather than the utility difference.

3.4 Properties of Subsidy Instability

We now describe additional properties satisfied by our instability measure that are important for learning. We show that Subsidy Instability is: (i) zero if and only if the matching with transfers is stable, (ii) Lipschitz in the true utility functions, and (iii) lower bounded by the utility difference.

Proposition 3.3 (Informal)**.** *Subsidy Instability satisfies the following properties:*

1. *Subsidy Instability is always nonnegative and zero if and only if (X, τ) is stable.*
2. *Subsidy Instability is Lipschitz continuous with respect to agent utilities.*
3. *Subsidy Instability is always at least the utility difference.*

These three properties show that Subsidy Instability is useful as a regret measure for learning stable matchings. The first property establishes that Subsidy Instability satisfies the basic desideratum of having zero instability coincide with exact stability. The second property shows that our measure of instability is robust to small perturbations to the utility functions of individual agents. The third property ensures that the platform learns a socially optimal matching when using Subsidy Instability as a loss function.

While property 2 implies the existence of an $\widetilde{O}(N^{4/3}T^{2/3})$-regret explore-then-commit algorithm for learning a stable matching, we will show that we can improve the dependence on the number of rounds to $\sqrt{T}$ in the next section.

4 Learning Stable Matchings in a Bandits Framework

In this section, we develop a general approach for designing algorithms that achieve near-optimal regret within our framework. The algorithm itself is simple to describe and follows the principle of "optimism in the face of uncertainty." Our approach composes with incentive-free approaches in a simple manner, without any asymptotic loss in performance.

Each round, the algorithm selects a stable matching with transfers using upper confidence bounds as estimates of true agent utilities. To design and analyze this algorithm, we leverage the fact that, in the full-information setting, stable matchings and prices are optimal solutions to a pair of primal-dual linear programs whose coefficients depend on the true utilities. This primal-dual perspective lets us compute a matching with transfers each round and analyze its suboptimality in terms of the sizes of the confidence sets. A particular consequence is that any UCB-style algorithm for learning matchings

Algorithm 1 COMPUTEMATCH: Compute matching with transfers from confidence sets

procedure COMPUTEMATCH($\mathscr{C}$)
  for $(i,j) \in \mathcal{I} \times \mathcal{J}$ **do**
    $u_i^{\text{UCB}}(j) \leftarrow \max\big(C_{u_i(j)}\big)$; $\quad u_j^{\text{UCB}}(i) \leftarrow \max\big(C_{u_j(i)}\big)$ ▷ UCB estimates of utilities.
  Compute optimal primal-dual solutions (X^*, p^*) to (P) and (D) for utility functions u^{UCB}.
  for $a \in \mathcal{A}$ **do**
    $\tau_a = p_a^* - u_a^{\text{UCB}}(\mu_{X^*}(a))$
  return (X^*, τ)

in a semi-bandit setting can be transformed into an algorithm for learning *both* the matching and the prices. In Section 4.3, we show this for the simplest structure on utility functions, and in Section 4.4, we generalize to linear preferences and matchings with non-transferable utilities.

4.1 Stable matchings via linear programming duality

We turn to the primal-dual framework for selecting a matching with transfers in the perfect information setting. Shapley and Shubik [30] show that stable matchings with zero-sum transfers (X, τ) correspond to optimal primal-dual solutions to the following pair of primal and dual linear programs:

Primal (P)

$$\begin{aligned} \max_{Z \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{J}|}} & \sum_{(i,j) \in \mathcal{I} \times \mathcal{J}} Z_{i,j}(u_i(j) + u_j(i)) \\ \text{s.t.} & \sum_{j \in \mathcal{J}} Z_{i,j} \le 1 \quad \forall i \in \mathcal{I} \\ & \sum_{i \in \mathcal{I}} Z_{i,j} \le 1 \quad \forall j \in \mathcal{J} \\ & Z_{i,j} \ge 0 \qquad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned}$$

Dual (D)

$$\begin{aligned} \min_{p \in \mathbb{R}^{|\mathcal{A}|}} & \sum_{a \in \mathcal{A}} p_a \\ \text{s.t.} & \; p_i + p_j \ge u_i(j) + u_j(i) \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \\ & \; p_a \ge 0 \qquad \forall a \in \mathcal{A} \end{aligned}$$

The primal program (P) is a linear programming formulation of the maximum weight matching problem: the Birkhoff-von Neumann theorem states that its extreme points are exactly the indicator vectors for matchings between $\mathcal{I}$ and $\mathcal{J}$. Each dual variable p_a in (D) can be interpreted as a *price* that roughly corresponds to agent a's net utility. Specifically, given any optimal primal-dual pair (Z, p), one can recover a matching μ_X from the nonzero entries of Z and set transfers $\tau_a = p_a - u_a(\mu_X(a))$ to obtain a stable outcome (X, τ). Moreover, any stable outcome induces an optimal primal-dual pair (Z, p). For completeness, we prove these facts in Appendix A.

4.2 Leveraging the primal-dual structure

We use this primal-dual perspective in two ways: first, when computing a matching with transfers at each round, and second, when analyzing its suboptimality.

In each round, we compute a matching with transfers by solving the primal-dual linear programs for our upper confidence bounds: Suppose we have a collection $\mathscr{C}$ of confidence sets $C_{u_i(j)}, C_{u_j(i)} \subseteq \mathbb{R}$ such that $u_i(j) \in C_{u_i(j)}$ and $u_j(i) \in C_{u_j(i)}$ for all $(i,j) \in \mathcal{I} \times \mathcal{J}$. Our algorithm uses $\mathscr{C}$ to get an upper confidence bound for each agent's utility function and then computes a stable matching with transfers as if these upper confidence bounds were the true utilities (see COMPUTEMATCH). This can be implemented efficiently if we use, e.g., the Hungarian algorithm [19] to solve (P) and (D).

The core property of this algorithm is that we can upper bound Subsidy Instability by the sum of the sizes of the relevant confidence sets, assuming that the confidence sets contain the true utilities. In other words, small errors in the utilities can be massaged into stable solutions with small subsidies.

Lemma 4.1. *Consider a collection confidence sets $\mathscr{C}$ such that $u_i(j) \in C_{u_i(j)}$ and $u_j(i) \in C_{u_j(i)}$ for all $(i,j) \in \mathcal{I} \times \mathcal{J}$. The instability of the output (X^{UCB}, τ^{UCB}) of* COMPUTEMATCH *satisfies*

$$I(X, \tau; u) \le \sum_{a \in \mathcal{A}} \Big(\max\big(C_{u_a(\mu_X(a))}\big) - \min\big(C_{u_a(\mu_X(a))}\big) \Big). \tag{3}$$

We prove this lemma from a dual perspective. Our goal is to select subsidies s such that $(X, \tau + s)$ is stable. Duality is useful here because dual-feasible solutions correspond to agent utilities obtained by stable matchings with transfers when the zero-sum transfers constraint is relaxed:

Proposition 4.2. *If p is dual feasible, then there exists a matching with transfers (X, τ) (for τ not necessarily zero-sum) such that (X, τ) is stable and $p_a = \tau_a + u_a(\mu_X(a))$. The converse also holds.*

To prove Lemma 4.1, we start from the dual solution p corresponding to (X, τ). That is, we take p to be such that $p_a = u_a(\mu_X(a)) + \tau_a$. (Note that p_a is not necessarily dual feasible.) By Proposition 4.2, we see that $(X, \tau + s)$ is stable if and only if $p' = p + s$ is dual-feasible. That $s \succeq 0$ implies $p' \succeq p$ (where $\succeq$ is taken to mean coordinate-wise inequality). Take $p' = p^*$ from COMPUTEMATCH. This p^* is dual feasible for the true utilities because it is dual feasible for the upper confidence bounds and the upper confidence bounds are at least the true utilities. Moreover, $p^* \succeq p$, so we can take the subsidies to be $s = p^* - p'$. Then,

$$s_a = \max\big(C_{u_a(\mu_X(a))}\big) - u_a(\mu_X(a)) \leq \max\big(C_{u_a(\mu_X(a))}\big) - \min\big(C_{u_a(\mu_X(a))}\big),$$

from which (3) follows. This choice of subsidies has a clean economic intuition: agents should be compensated based on the platform's uncertainty about their utilities.

Remark. The platform could also set $s_a = \max(C_{u_a(\mu_X(a))}) - \min(C_{u_a(\mu_X(a))})$. This choice of s has the benefit that it can be computed using only knowledge that the platform has, meaning that the platform could pay agents these subsidies and realize the regret as a "cost of learning."

4.3 Explicit algorithm and regret bounds

The regret bound of Lemma 4.1 hints at an algorithm: each round, select the matching with transfers returned by COMPUTEMATCH and update confidence sets accordingly. To instantiate this approach, it remains to construct confidence intervals that contain the true utilities with high probability. This last step naturally depends on the assumptions made about the utilities and the noise.

For the remainder of this subsection, we consider the following simple setting: All agents' utilities lie in $[-1, 1]$ and the noise variables $\varepsilon_{a,t}$ are independent, 1-subgaussian random variables. For this setting, we may construct our confidence intervals following the classical UCB approach. That is, for each utility value involving the pair (i, j), we take a length $O(\sqrt{\log(|\mathcal{A}|T)/n_{ij}})$ confidence interval centered around the empirical mean, where n_{ij} is the number of times the pair has been matched before. We describe this construction precisely in Algorithm 2 (MATCHUCB).

Algorithm 2 MATCHUCB: A bandit algorithm for matching with transferable utilities.

procedure MATCHUCB(T)
  for $(i, j) \in \mathcal{I} \times \mathcal{J}$ **do** ▷ Initialize confidence intervals and empirical mean.
    $C_{u_i(j)} \leftarrow [-1, 1]$; $C_{u_j(i)} \leftarrow [-1, 1]$; $\hat{u}_i(j) \leftarrow 0$; $\hat{u}_j(i) \leftarrow 0$
  for $1 \leq t \leq T$ **do**
    $(X^t, \tau^t) \leftarrow$ COMPUTEMATCH(C)
    for $(i, j) \in X^t$ **do** ▷ Set confidence intervals and update means.
      Update $\hat{u}_i(j)$ and $\hat{u}_j(i)$ from feedback; increment counter n_{ij}
      $C_{u_i(j)} \leftarrow \big[\hat{u}_i(j) - 8\sqrt{\log(|\mathcal{A}|T)/n_{ij}}, \hat{u}_i(j) + 8\sqrt{\log(|\mathcal{A}|T)/n_{ij}}\,\big]$
      $C_{u_j(i)} \leftarrow \big[\hat{u}_j(i) - 8\sqrt{\log(|\mathcal{A}|T)/n_{ij}}, \hat{u}_j(i) + 8\sqrt{\log(|\mathcal{A}|T)/n_{ij}}\,\big]$

To analyze MATCHUCB, recall that Lemma 4.1 bounds the regret at each step by the lengths of the confidence intervals of each pair in the selected matching. Bounding the lengths of the confidence intervals parallels the analysis of UCB for classical stochastic multi-armed bandits and yields the following instance-independent regret bound:

Theorem 4.3. MATCHUCB *incurs expected regret* $\mathbb{E}(R_T) \leq O\big(|\mathcal{A}|^{3/2}\sqrt{T}\sqrt{\log(|\mathcal{A}|T)}\big)$.

Revisiting the economic motivation, Theorem 4.3 tells us that the average subsidy is $\widetilde{O}\big(\sqrt{|\mathcal{A}|/T}\big)$ per user per time step. While this does depend on the number of agents, we show in Section 4.4 that this dependence on the number of agents can removed (up to logarithmic factors) with appropriate structure on the preferences. Even without assuming any structure on the preferences, the platform can still achieve subconstant average subsidies per agent after $T = \Theta(|\mathcal{A}|)$ rounds.

Matching lower bound. We now show that MATCHUCB achieves optimal regret (up to logarithmic factors) by showing a lower bound that (nearly) matches the upper bound in Theorem 4.3.

Lemma 4.4. *For any algorithm for learning a matching with transfers, there exists an instance on which it has expected regret $\widetilde{\Omega}(|A|^{3/2}\sqrt{T})$ (where regret is given by Subsidy Instability).*

The idea behind this lemma is to show a lower bound for the easier problem of learning a maximum matching. By Proposition 3.3, this immediately implies a lower bound for learning a stable matching with regret measured by Subsidy Instability.

This lower bound illustrates the close connection between our setting and that of learning a maximum matching. Indeed, by applying MATCHUCB and simply disregarding the transfers every round, we recover the classical UCB-based algorithm for learning the maximum matching [15, 9, 20]. From this perspective, the contribution of MATCHUCB is an approach to set the dual variables while asymptotically maintaining the same regret as the primal-only problem.

4.4 Extensions

Instance-dependent regret Applying a similar analysis as that of Chen et al. [8] for combinatorial bandits, we obtain an $O(\log T)$ instance-dependent regret bound for MATCHUCB. As is typical in combinatorial bandits problems (e.g., [20, 8]), the hidden constant in our bound depends on a gap Δ that is global to the matching. We formalize this in Appendix D.

Linear preferences. While we have focused thus far on unstructured agent preferences, a feature of our framework is that it composes nicely with existing UCB analyses in more general contexts. To show this, we extend our results to linear preference structures. Here each agent a has an associated r-dimensional feature vector $f_a \in \mathbb{R}^r$, and for a and a' on opposite sides of the market, we define $u_a(a') = \langle f_a, f_{a'} \rangle$ (we extend this to asymmetric preferences in Appendix E). We assume that the feature vectors for the providers are known to the platform ahead of time, whereas the feature vectors for the customers must be learned. This low-dimensional structure enables us to shrink the size of the confidence sets: in the classical multi-armed bandits setting, LinUCB [21] takes advantage of these correlations to build ellipsoidal confidence sets.

We show that the same high-level approach as in Section 4.3 of using the UCB estimates to select a matching with transfers (but with confidence sets adapted from LinUCB) applies to this setting. We can then combine Lemma 4.1 with the existing analysis of LinUCB to get a $\widetilde{O}(r|\mathcal{A}|\sqrt{T})$ regret bound. In particular, the average instability incurred per agent is $\widetilde{O}(r\sqrt{T})$. We can also extend this analysis to *generalized linear preferences* [27]. We formalize these results in Appendix E.

Matchings with non-transferable utilities. While we have focused on matching with transferable utility, utilities are not always transferable in practice, as in the case of dating markets or college admissions. We can extend our findings to the classical setting of matchings *without* transferable utilities [16], which has also been studied in previous work [11, 22, 6, 28]. The definition of Subsidy Instability extends naturally and has advantages over the "utility difference" metric of previous work. Our algorithmic meta-approach also sheds new light on the convergence properties of the centralized UCB algorithm of Liu et al. [22]. We discuss this in more detail in Appendix F.

5 Discussion

We have introduced a framework for learning equilibria (i.e., stable outcomes) in matching markets from bandit feedback. A core component of this framework is a new measure of the *instability* of a matching outcome that takes into account both the quality of the allocation and the accuracy of the prices. By using instability as a loss function, our learning algorithms designed are *incentive-aware*—they optimize not only for the aggregate objective (i.e., social welfare), but also the satisfaction of individual agents. Algorithmically, our framework composes cleanly with the bandits literature, with classical approaches such as UCB extending naturally to our setting. Our framework extends naturally to the setting where utilities are non-transferable.

In terms of broader impacts, one limitation is that we focus on equilibrium outcomes and do not address potential differences in utilities across users. It would be interesting to incorporate fairness criteria, potentially drawing inspiration from fairness in other mechanism design problems [13].

References

[1] Guy Aridor, Yishay Mansour, Aleksandrs Slivkins, and Zhiwei Steven Wu. Competing bandits: The perils of exploration under competition. *CoRR*, abs/2007.10144, 2020.

[2] Itai Ashlagi, Mark Braverman, Yash Kanoria, and Peng Shi. Clearing matching markets efficiently: informative signals and match recommendations. *Management Science*, 66(5): 2163–2193, 2020.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[4] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *J. ACM*, 65(3):13:1–13:55, 2018.

[5] Soumya Basu, Karthik Abinav Sankararaman, and Abishek Sankararaman. Beyond $\log^2(t)$ regret for decentralized bandits in matching markets. *CoRR*, abs/2103.07501, 2021.

[6] Sarah H. Cen and Devavrat Shah. Regret, stability, and fairness in matching markets with bandit learners. *CoRR*, abs/2102.06246, 2021.

[7] Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78(5): 1404–1422, 2012.

[8] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 151–159. JMLR.org, 2013.

[9] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *CoRR*, abs/1407.8339, 2014.

[10] Richard Combes, Mohammad Sadegh Talebi, Alexandre Proutière, and Marc Lelarge. Combinatorial bandits revisited. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 2116–2124, 2015.

[11] Sanmay Das and Emir Kamenica. Two-sided bandits and the dating market. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 947–952. Professional Book Center, 2005.

[12] Ehsan Emamjomeh-Zadeh, Yannai A. Gonczarowski, and David Kempe. The complexity of interactively learning a stable matching by trial and error. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation*, page 599. ACM, 2020.

[13] Jessie Finocchiaro, Roland Maio, Faidra Monachou, Gourab K. Patro, Manish Raghavan, Ana-Andreea Stoica, and Stratis Tsirtsis. Bridging machine learning and mechanism design towards algorithmic fairness. In Madeleine Clare Elish, William Isaac, and Richard S. Zemel, editors, *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 489–503. ACM, 2021.

[14] Peter I. Frazier, David Kempe, Jon M. Kleinberg, and Robert Kleinberg. Incentivizing exploration. In Moshe Babaioff, Vincent Conitzer, and David A. Easley, editors, *ACM Conference on Economics and Computation, EC '14*, pages 5–22. ACM, 2014.

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

[16] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[17] Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. *Games Econ. Behav.*, 118:626–647, 2019.

[18] Robert D. Kleinberg and Frank Thomson Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 594–605. IEEE Computer Society, 2003.

[19] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[20] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvári. Tight regret bounds for stochastic combinatorial semi-bandits. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[21] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

[22] Lydia T. Liu, Horia Mania, and Michael I. Jordan. Competing bandits in matching markets. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1618–1628. PMLR, 2020.

[23] Lydia T. Liu, Feng Ruan, Horia Mania, and Michael I. Jordan. Bandit learning in decentralized matching markets. *CoRR*, abs/2012.07348, 2020.

[24] Yishay Mansour, Aleksandrs Slivkins, and Vasilis Syrgkanis. Bayesian incentive-compatible bandit exploration. In Tim Roughgarden, Michal Feldman, and Michael Schwarz, editors, *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15*, pages 565–582. ACM, 2015.

[25] Maria Silvia Pini, Francesca Rossi, and Kristen Brent Venable. Stable matching problems with soft constraints. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14*, pages 1511–1512. IFAAMAS/ACM, 2014.

[26] Michael Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9(2):185–202, 1974.

[27] Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 2256–2264, 2013.

[28] Abishek Sankararaman, Soumya Basu, and Karthik Abinav Sankararaman. Dominate or delete: Decentralized competing bandits in serial dictatorship. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1252–1260. PMLR, 2021.

[29] L. S. Shapley and M. Shubik. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica*, 34(4):805–827, 1966.

[30] L. S. Shapley and M. Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1(1):111–130, December 1971.

[31] Peng Shi. Efficient matchmaking in assignment games with application to online platforms. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation*, pages 601–602. ACM, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] (See Section 5.)
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] (See supplementary material.)
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]