DS 102 Lecture 20: Matching Markets

Lecturer: Michael I. Jordan

November 3, 2020

1 Two-Sided Markets

We consider situations where we want to model interactions between two sets of entities. One prominent example occurs in kidney exchange programs, where kidney donors and receivers need to be matched based on factors such as blood type compatibility. This is an example of a **matching market**, where the needs of one party need to be matched to the services provided by another party, typically in a way that accounts for the preferences of both sides. The platforms provided by Uber and Lyft are another example, in which passengers need to be matched to drivers. Another important example is the process of matching medical school students to hospitals, where both sides of the markets have preferences: each student has preferences over the hospitals, based on their interests, and each hospital has preferences over the students, based on their needs.

In this lecture, we'll use the running example of matching bands to drummers. Consider a set of m bands and a setv of n drummers. Every band wants a drummer, and every drummer wants to belong to a band. Each band has preferences over the different drummers, and each drummer has preferences over the different bands. We model these preferences as a total order for each drummer and each band, and we assume that these preferences are known and fixed. An example of the preferences of a set of bands and a set of drummers is given in Fig. 1.

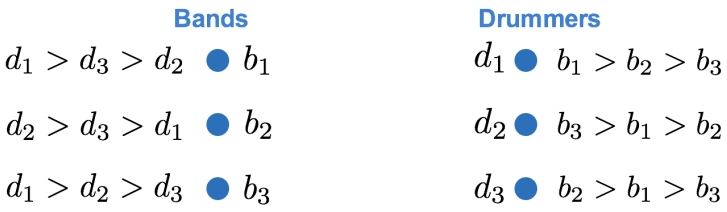

Figure 1: Example preferences among $m = 3$ bands and $n = 3$ drummers. For example, band b_1 prefers drummer d_1 over drummer d_3, and prefers drummer d_3 (and d_1) over drummer d_2.

Given an arbitrary preferences, is it possible to find a matching where everyone is satisfied (in some sense)? We can also expanding out thinking to consider situations where either side doesn't know their preferences *a priori*, and has to learn them on the fly. This is a more modern take on the problem and an open area of research, and connects back to ideas from our unit on bandits.

2 Stable Matching Problem

To formalize the problem we'd like to solve, we'll start with some definitions.

Definition: A **matching** M is a set of pairs (b, d) such that each band b appears in at most one pair in M, and each drummer d appears in at most one pair M.

Another way to say this is that a matching is a one-to-one mapping between a subset of bands to a subset of drummers. By convention, if either a band or a drummer is not matched with anyone, they are considered matched with themselves. We assume that everyone prefers to be matched to someone else to being matched to themselves.

2.1 Stability

To specify what we're looking for in a "good" matching, intuitively, if a matching is satisfactory to everyone then it should obey some sort of equilibrium. To make this notion precise, we define a **blocking pair** and **unstable** and **stable** matchings.

Definition: Given a particular matching M, a **blocking pair** (b, d) is a pair such that b prefers d to its current match in M and d prefers b to its current match in M.

Definition: If a matching M has a blocking pair, it is **unstable**. Else, it is a **stable** matching.

This concept of stability is a local one, not a global one. That is, to be stable, a matching does not have to be compared to all possible matchings that could exist. It simply cannot contain a pair of a band and a drummer who would both prefer to be matched to each other than their current matches. Note that in a stable matching, any band or drummer can still have envy (preferring to be matched to a different drummer or band, respectively), as long as that envy is not "reciprocal". This definition of stability characterizes a sort of equilibrium, because there is no unmatched pair of a band and a drummer who would both benefit by being matched to each other instead.

3 Gale-Shapley Algorithm

Given arbitrary preferences, can we always find a stable matching? The answer is *yes*! In 1962, David Gale and Lloyd Shapley demonstrated that a simple, intuitive algorithm will always achieve this. The algorithm considers two sides of

the market, proposers and proposees. A key aspect of the algorithm is that proposers and proposees have different roles; the former actively proposes matches, while the latter passively accepts or rejects them. The algorithm can be applied with either group in either role in either way. For example, we could run it with the bands acting as proposers and the drummers acting as proposees (as we assume throughout today's lecture), or vice versa.

3.1 Algorithm

- Initialize M to an empty matching.
- While some band b is unmatched and hasn't proposed to every drummer:
  - Let d denote b's most preferred drummer, among all drummers to whom b has not yet proposed.
  - If d is unmatched, $M \leftarrow M \cup (b, d)$.
  - Else if d prefers b to b' (d's current match), $M \leftarrow (M \setminus (b', d)) \cup (b, d)$.
  - Else d rejects b, and M is unchanged.

At some point this algorithm will terminate, because there are only a finite number mn of possible proposals made by a band to a drummer, and each of these proposals can only be made once. After this finite number of proposals is made, then the algorithm terminates. Until it terminates, any matches made (*i.e.*, proposals by a band that are accepted by a drummer) are tentative, and could change in future rounds if the drummer receives a proposal from a band they prefer more. In fact, this is a key observation about the algorithm: a drummer's match can only improve as the algorithm proceeds.

3.2 Proof that Gale-Shapley Produces a Stable Matching

We can prove that the Gale-Shapley algorithm creates a matching M with no blocking pairs. To do this, we will show that for any pair (b, d) that is not a match in M, (b, d) is not a blocking pair. There are two cases to consider:

Case 1: If b never proposed to d during the execution of the algorithm, then we know that b prefers its match in M over d, because b proposed in order of its preferences. That is, the algorithm must have stopped before b got to proposing to d. So (b, d) cannot be a blocking pair, since b does not prefer d over its current match in M.

Case 2: If b did propose to d, then we know that d prefers its match in M over b, because (as observed above) the drummer d can only improve their match when they accept new proposals throughout the course of the algorithm. So (b, d) cannot be a blocking pair, since d does not prefer b over its current match in M.

We have therefore shown that any pair that is not a match in M cannot be blocking. Therefore, the matching M produced by the Gale-Shapley algorithm is stable.

4 Optimality and Pessimality of the Gale-Shapley Matching

A problem instance can have multiple stable matchings, which are better or worse for different members of the market depending on their preferences. We will show that Gale-Shapley provides the proposers (here, the bands) the optimal matching from their point of view. However, it provides the worst matching for drummers, among stable matchings. To make this more precise, we first define **attainability**.

Definition: b and d are **attainable** for each other if there exists a stable matching in which b and d are matched.

We can now state the main result describing the particular stable matching found by Gale-Shapley.

Theorem: The matching produced by the Gale-Shapley algorithm matches each proposer with their best attainable partner, and matches each proposee with their worst attainable partner.

In other words, Gale-Shapley is **proposer-optimal** and **proposee-pessimal**.

4.1 Proof of Proposer-Optimality

We will do a proof by contradiction that Gale-Shapley is proposer-optimal. The proof for proposee-pessimality is similar, but we won't worry about going through it here.

Note that since each band proposes in order of their preferences, every band being matched to their best attainable drummer means that no band is every rejected by an attainable drummer. Therefore, to start the proof by contradiction, we assume that at some point there is a band b that is rejected by an attainable drummer d. Let b be the first band rejected by an attainable match d. Then at the time of the rejection, there must have been another band b' that was tentatively matched to d that d preferred over b. Also, b' must prefer d among all of its attainable drummers, because b' wasn't ever rejected by an attainable drummer before it proposed to d (since we define b as the first band to be rejected by an attainable drummer).

The first part of the proof simply used definitions to help us infer properties of the preferences of d, b, and b'. We will use these properties to now wrap up the proof. Because d is attainable for b (by assumption), by definition there exists some stable matching M' in which they are matched. Since b and d are matched in M', b' must matched to some other d', instead of matching to d. This means (b', d) is a blocking pair in M': we've deduced above that b' prefers d over all attainable drummers, including d', and we also deduced above that d prefers b' over b. We have a contradiction, as we've now shown that M' is not a stable matching! Therefore, the initial assumption that there is some band that is rejected by an attainable drummer must be wrong. We conclude that every band always gets their most preferred attainable drummer.

5 Learning in Matching Markets

What if the preferences are not known *a priori*? For example, bands may not know about drummers' skills and styles and how suitable they are for the band. We could consider using a bandit algorithm like UCB to simultaneously learn one's preferences while proposing. This produces a situation of multiple bandit-based learners (for example, the bands) interacting with the different arms (for example, drummers). If, in a given round, multiple learners choose the same bandit arm, scarcity—a key concept from economics—kicks in: only one of them gets the reward, and the others get no reward. Which one gets the reward? The one that the arm prefers! (Here, we assume the arm's preferences are fixed and arbitrary, but we could also extend this problem to arms learning their preferences.) So the learner without a reward gets an extra piece of information that the arm prefers that other learners, and may decide it's not worth pursuing this arm anymore. (Or, in contrast, the learner that receives the reward has evidence that the arm prefers it over other competitors, and may decide to aggressively pursue this arm.) As in our unit on bandits, we a trade-off between exploration and exploitation.

One application of this problem is finding the fastest route to a popular destination, like an airport. If too many drivers try a short route, there will be congestion and most of the drivers' rewards will go down. Similarly, in matching diners to restaurants, diners want to explore and learn what the best restaurants are, but due to limited capacity, the more diners that try a good restaurant the harder it is to get a reservation.