CS 161 (Stanford, Winter 2025) Lecture 17

Adapted from lecture notes by Aviad Rubinstein and modified by Nima Anari.
Please direct all typos and mistakes to Moses Charikar and Nima Anari.

Stable Matching and Gale-Shapley

Every year, tens of thousands of doctors begin their residencies in the United States. The process of matching doctors to their residencies is extremely complex, and must take into account the doctors' preferences and the needs of the hospitals. In this lecture we will learn the **Deferred Acceptance Algorithm** which is at the core of *The Match*, a.k.a. the National Resident Matching Program. The algorithm is sometimes also called the *Gale-Shapley* algorithm after its inventors, David Gale and Lloyd Shapley.

1 Stable matching

Setting: Our goal is to match a collection of doctors and a collection of hospitals to each other. Throughout, we assume that each hospital has exactly one position to fill. This is just to simplify the verbiage and notation, and does not lose generality; in the more general case where hospitals have larger capacity, replace the word "hospital" by "hospital position" in the ensuing text and everything still goes through. We also assume the number of hospitals and doctors is the same and we call that n. This is again to simplify the notation; when these numbers are not equal, some doctors (or hospitals) might have to remain unmatched. We can model this by adding fake hospitals (or doctors) that are ranked very low, and matching to them counts as being "unmatched" in real life.

Suppose that we had some numerical score for each pair of doctor i and hospital j hinting at how good of a fit pairing i and j is. Then, we may try to design an algorithm that finds an optimal (say max-weight) matching in this graph.

One problem with this approach is that we do not have numerical scores, so we have to rely on information from doctors and hospitals themselves. We might ask each doctor to rank the hospitals, and each hospital to rank the doctors. But even then, it is not straightforward how to translate the preferences of both a doctor and a hospital, two rankings, into a single numerical score. Any such scheme can also be prone to "gaming" by the doctors and the hospitals, because they can possibly manipulate the outcome and gain a better match by reporting false preferences.

Even worse, in order to use a centralized matching algorithm, you must convince thousands of residency programs to list their positions on your algorithm and commit to taking the doctors you assign them. So you better ensure that hospitals can't find better matches outside your algorithm!

This motivates the definition of a stable matching:

Definition 1 (Stable Matching)**.** We say that a bipartite matching between doctors and hospitals is *stable* if for every doctor i and hospital j, one of the following happens:

- i, j are matched to each other; or
- Hospital j is matched to another doctor which it prefers over i; or
- Doctor i is matched to another hospital which it prefers over j.

Intuitively, if doctor i prefers Hospital j to its match in the algorithm, and hospital j prefers doctor i, then they would jointly prefer to find their match without your algorithm. Such a pair (i, j) is called a *blocking pair*, and a matching is stable if it does not admit any blocking pairs.

So finding a stable matching resolves the problem of doctors and hospitals trying to bypass your algorithm. (There is still the issue of misrepresenting preferences. We will get back to this issue in Section 4.)

Example 2. Consider three doctors $\{\text{Alice}, \text{Bob}, \text{Charlie}\}$, and three hospitals $\{X, Y, Z\}$.

Suppose that Alice prefers Y over X over Z, etc:

Table 1: Doctor preferences

Doctor	Alice	Bob	Charlie
top choice	Y	X	X
2nd choice	X	Y	Y
last choice	Z	Z	Z

On the hospital side, their preferences are:

Table 2: Hospital preferences

Hospital	X	Y	Z
top choice	Alice	Charlie	Bob
2nd choice	Charlie	Alice	Charlie
last choice	Bob	Bob	Alice

- Observe that the matching

$$\text{(Alice-X),(Bob-Z),(Charlie-Y)} \tag{1}$$

  is stable: every hospital gets its favorite doctor, so no hospital wants to hire a doctor outside the matching. (The doctors are less happy about it...)

- Another stable matching is

$$\text{(Alice-Y),(Bob-Z),(Charlie-X).} \tag{2}$$

- The matching

$$\text{(Alice-Z),(Bob-X),(Charlie-Y)} \tag{3}$$

  is *not* stable, because Alice and X prefer each other over their respective matches.

2 The algorithm

The **Deferred Acceptance Algorithm** is essentially based on a greedy strategy: at each step we try to match each doctor to her most preferred hospital, except for hospitals that we already have tried and know are impossible.

There is a twist which distinguishes this algorithm from other greedy algorithms we've seen previously in the course. The algorithm's greedy choices are revocable, i.e., the matches are not final until the end of the algorithm. Even if, for example, doctor i matched with hospital j at the first round of the algorithm, if in a later iteration, doctor k also wants to match with hospital j, we will break ties between the doctors based on hospital j's preferences. In particular, if hospital j prefers k over i, then the previous match between j and i is canceled, and doctor i is sent to find a new hospital.

Algorithm 1: Gale-Shapley, a.k.a. the Deferred Acceptance Algorithm

input : $n \times n$ matrices D and H. The i-th row of D is the list of hospitals in the order of most preferred to least preferred from doctor i's view. $H[j][i]$ is the rank of doctor i from hospital j's perspective (lower number means more preferred).

output : Matching between doctors and hospitals.

```
                                                    // Initialization
```

for $d \in$ *doctors* **do**
  $d.i \leftarrow 0$

for $h \in$ *hospitals* **do**
  h.doctor $\leftarrow$ NIL

freeDoctors $\leftarrow$ doctors

```
                          // While there is a free doctor we keep matching.
```

while *freeDoctors* $\neq \emptyset$ **do**
  Pick any $d \in$ freeDoctors
  `// h is the next preferred hospital for d.`
  $d.i \leftarrow d.i + 1$
  $h \leftarrow D[d][d.i]$

```
                                  // If h prefers d over previous match, switch.
```

  if $H[h][d] < H[h][h.doctor]$ **then**
    Add h.doctor to freeDoctors
    Remove d from freeDoctors
    h.doctor $\leftarrow d$

return *the matching* $\{(h.doctor, h) \mid hospitals\ h\}$

3 Proof of correctness

In this section, we will prove that Algorithm 1 always outputs a stable matching. Let us start by proving some observations about the behavior of Algorithm 1.

Proposition 3. *If h.doctor becomes not NIL, it will never become NIL again.*

Proof. We only assign NIL values at initialization. So once a hospital has a *potential match* it can never run out of matches again; it will only reject a potential match for another match (a *more preferred* one). □

Proposition 4. *In Algorithm 1, no doctor "runs out of" hospitals to try. In other words $d.i$ never becomes $n+1$.*

Proposition 5. *If a doctor d were to run out of hospitals to try, it means that they must have already tried all hospitals. But this means every hospital already has a potential match; at the time the hospital rejected d, they must have already had another match, and by the previous proposition, the hospital's match never becomes NIL from that point onwards.*

But now we get a contradiction. There are n hospitals, each of which has a current match other than d. So all of the current hospital matches must come from the set of $n-1$ doctors other than d, which is impossible.

Proposition 6. *The algorithm terminates in at most $O(n^2)$ iterations and returns a matching.*

Proof. In every iteration, some $d.i$ gets incremented by 1. Since they start at 0 and can never reach $n+1$, in total the number of increments is at most $n \times O(n) = O(n^2)$. By the previous proposition, when the algorithm terminates, we have a full matching. □

Now that we know Algorithm 1 terminates and returns a matching, it only remains to prove that the matching is **stable**.

Theorem 7. *The matching returned by Algorithm 1 is stable.*

Proof. First note that a hospital's matches only improve over the course of the algorithm. In other words, for every h, h.doctor's ranking according to h can only get better and better as we run through the algorithm. This is because h will only reject a potential match for a better one.

Let us call the matching returned by Algorithm 1, σ. Assume by contradiction, that there is a blocking pair (i, j) for σ. This means that i prefers j to her match in σ and j prefers i to its match in σ. Let h be the hospital that i got matched to in σ. Since doctors go over their preference list one-by-one in Algorithm 1 and h comes later in the list of i than j, this means at some point during the run of the algorithm j must have been "proposed to" by i. So j had the opportunity to keep i or potentially reject i for better and better matches. But j ended up with a match that's worse than i; this is a contradiction, since j's matches must have only gotten better throughout the algorithm. □

4 Doctor-optimality and incentive compatibility

Let σ be a stable matching. We say that it is *doctor-optimal* if for any other stable matching σ' and any doctor d, the doctor d weakly prefers her hospital in σ to her hospital in σ'. (*Weakly prefer* means that it is either better or the same.) Note that a priori it is not clear that doctor-optimal stable matchings even exist.

In Example 2, the doctor-optimal matching is given in Eq. (2), and Eq. (1) is the stable matching that hospitals prefer. This is not a coincidence: the doctor-optimal stable matching is provably always the hospital-worst stable matching.

Theorem 8 (Doctor-optimality)**.** *The output of Algorithm 1 is* doctor-optimal*.*

Proof. For each doctor d, let $h^*(d)$ be her best feasible hospital, i.e,. the one she prefers the most among any hospital she can be matched to in any stable matching. Assume *by contradiction* that Algorithm 1 is not doctor-optimal. This means some doctor d gets rejected by her best possible choice $h^*(d)$ at some point during the algorithm. Consider the first time during the run of Algorithm 1 where this happens. This means $h^*(d)$ rejects d because a new doctor d' tries to match with it and $h^*(d)$ prefers d' over d. Now there are two cases:

- $h^*(d') = h^* = h^*(d)$, i.e., they both have the same best feasible hospital h^*. Therefore Hospital h^* will always prefer d' over d, and d' always prefers it to its match in any stable matching. Therefore, it is never stable to match h^* to d (d' and h^* would be a blocking pair). But this is a contradiction to h^* being feasible for d!
- $h^*(d') \neq h^*(d)$, in which case doctor d' should have first tried to match with $h^*(d')$ before coming to $h^*(d)$. If d' tries to match with $h^*(d)$ instead, it must have been rejected by $h^*(d')$. But this is a contradiction to d being the first doctor to be rejected by her best feasible hospital!

□

Incentive compatibility. Consider a doctor that strategically misreports her preferences in hope of a better hospital. Intuitively, given the new (misreported) preferences, Algorithm 1 should still output a stable matching. By the doctor-optimality theorem that we just proved, the doctor is already assigned the best match she can possibly be assigned in any stable matching; hence misreporting preferences cannot help her. This logic is actually flawed (can you see why?[1]), but the conclusion is still true, namely no doctor has an incentive to misreport her preferences:

Theorem 9 (Incentive-compatibility)**.** *Misreporting the preference over hospitals can never improve a doctor's match computed by Algorithm 1.*

[1] Given the misreported preferences, Algorithm 1 should still output a matching that is stable *with respect to the misreported preferences*. But the doctor-optimality theorem only guaranteed that the output is doctor-optimal among all matchings that are stable with respect to the true preferences.

The correct proof of this theorem is beyond the scope of these notes. It can be found, e.g., in this research paper by Dubins and Freedman: `https://www-jstor-org.stanford.idm.oclc.org/stable/2321753`.

As a remark, unlike doctors, hospitals can actually misreport and potentially get better matches. To appreciate the incentive compatibility for the doctors, let us revisit Example 2, and see how hospitals can strategically misreport their preferences to improve their matching.

Example 10. Suppose hospital X misreports its preferences as in Table 3 (but doctors' preferences are still the same).

Table 3: Hospital mis-reported preferences

Hospital	X	Y	Z
top choice	Alice	Charlie	Bob
2nd choice	**Bob**	Alice	Charlie
last choice	**Charlie**	Bob	Alice

Now Eq. (1) is the unique stable matching, so the Deferred Acceptance Algorithm will choose it, and not Eq. (2) (can be verified by running the algorithm).