CS269I: Incentives in Computer Science
Lecture #2: Stable Matching*

Tim Roughgarden†

September 28, 2016

1 Stable Matching and the Deferred Acceptance Algorithm

Last lecture, we turned a critical eye to the way college admissions are run in the U.S. Could the system be improved? For example, could there be a sensible centralized clearinghouse, like a two-sided analog of the Draw? If so, what would it look like? For inspiration, we are looking at the National Resident Matching Program (NRMP), who had to solve a similar problem to assign newly minted medical school graduates to residency programs. We explained the history of the NRMP last time, where in 1952 a committee of medical school students, as an act of protest, proposed a matching procedure to replace the one the was being considered. Their alternative procedure was indeed adopted, and to a large extent survives to this day.[1] This algorithm and its properties are the subject of this lecture.

1.1 The Model

Consider a set S of students and a set H of hospitals. Each student has a ranked list of preferences over hospitals, and each hospital a ranked list over students. Each hospital $h \in H$ has a "capacity" c_h, indicating the number of positions that it has available. S and H constitute the two sides of the market; if you like, you can think of them as the two vertex sets of a bipartite graph. For example, in Figure 1, the students (on the left) have a common ranking of the hospitals, while the hospitals have very different opinions about the students.

For simplicity, we'll assume that $|S| = \sum_{h \in H} c_h$ — that the number of positions is exactly the number of applicants. It's easy to enforce this condition by adding "fake" students or hospitals (which are less preferred than any actual students or hospitals).

*©2016, Tim Roughgarden.

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: `tim@cs.stanford.edu`.

[1] Back in 1952, computers barely existed. Initially, the algorithm with implemented using card-sorting machines!

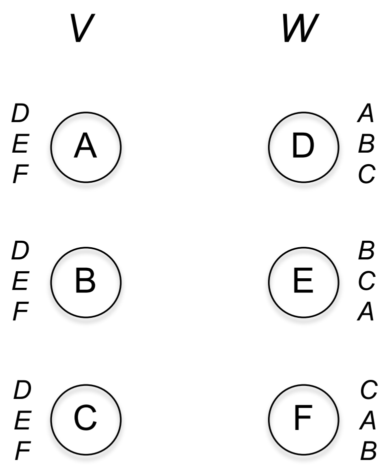

Figure 1: An instance of stable matching. Each vertex is annotated with its total ordering over the vertices of the opposite side, with the most preferred vertex on top. All hospitals have capacity 1.

By a *matching*, we mean an assignment of each student to exactly one hospital, so that each hospital h is assigned exactly c_h students. Here is the key definition.

Definition 1.1 (Stable Matching [2]) A matching of students to hospitals is *stable* if there is no pair $s \in S$ and $h \in H$ such that all of the following properties hold:

(i) s is not assigned to h;

(ii) s ranks h higher than the hospital she was assigned to;

(iii) h ranks s higher than one the c_h students assigned to it.

A student-hospital pair that satisfies (i)–(iii) is called a *blocking pair*. Thus a matching is stable if and only if there is no blocking pair. A blocking pair spells trouble, because the pair may be tempted to secede from the process and match with each other directly. (E.g., if you get assigned to your third-favorite hospital, it's not that hard to call your top two choices to double-check that they don't prefer you to any of the students they admitted. And if one of them does, both sides are motivated to work something out. . .)

For example, in Figure 1, consider any matching where A is not assigned to D. By assumption, (i) holds. Since A ranks D first and vice versa, S must be assigned to a hospitals she likes less than D (so (ii) holds) and H is assigned a student it likes less than A (so (iii) holds). Thus (A, D) form a blocking pair whenever A is not assigned to D, and every stable matching must match A and D.

Stability seems like a nice property — but can it always be achieved? A priori, it is not obvious that a stable matching is guaranteed to exist (for any preferences of the participants). In the example in Figure 1, we have already argued that any stable matching must assign

A to D. Check that assigning B to E and C to F yields a stable matching. (In fact, the unique one in this example.) Note that not everyone is happy in a stable matching — in the example, C is assigned to her least favorite hospital, but unfortunately for C, the other two hospitals prefer the students they were assigned.

What about in general? We next prove the existence of a stable matching through an (efficient) algorithm.

1.2 The Deferred Acceptance Algorithm

We next discuss the elegant deferred acceptance algorithm for computing a stable matching. This algorithm was first published by Gale and Shapley in 1962 [2], although the NRMP implemented an equivalent algorithm already ten years earlier (see [3]).

Deferred Acceptance Algorithm

```
while there is at least one unassigned student do
    every unassigned student "proposes" to her favorite hospital that
      hasn't rejected her yet
    every hospital keeps track of its favorite $c_h$ proposals that it ever
      receives, and rejects all other proposals[2]
all unrejected proposals are made final
```

And that's it! Note that not too much computation is required; the algorithm is easy to implement in $O(|S| \cdot |H|)$ time.

Example 1.2 (The Deferred Acceptance Algorithm) Consider the instance in Figure 1. In the first iteration, all of the students propose to D, who retains its favorite proposal (from A) and rejects the other two. In the second iteration, students B and C are still unassigned, and both propose to hospital E, who retains the higher-ranked proposal (from E) and rejects F. In the last iteration, C has no choice but to propose to F, and F has no choice but to accept it. Thus the algorithm terminates with the stable matching identified earlier.

1.3 Computation and Existence of Stable Matchings

We next note several properties of the deferred acceptance algorithm. First, each student systematically goes through her preference list, from top to bottom. Second, because a hospital only rejects an applicant in favor of a better one (and students never withdraw), the students to whom a hospital is tentatively matched only improve over the course of the algorithm. By the same reasoning, once a hospital becomes full in the algorithm, it remains full forevermore. Third, at all times, each student is matched to at most one hospital and each hospital h is matched to at most c_h students.

[2]If a student's proposal is initially tentatively accepted (i.e., not rejected), it still might get rejected in a later iteration of the algorithm. Can you find an example of this?

Stable matchings and the deferred acceptance algorithm have an astonishing number of remarkable properties.[3] Here is the most important one.

Theorem 1.3 ([2]) *The deferred acceptance algorithm always terminates with a stable matching.*

Proof: For starters, we claim that the deferred acceptance algorithm always terminates with every student matched to some hospital and c_h students matched to each hospital $h \in H$. For if not, some student must have been rejected by all of the hospitals. A student is only rejected by a hospital h in favor of being matched to c_h better students, and once a hospital is full (i.e., tentatively assigned c_h students), it remains so for the remainder of the algorithm. Thus, all hospitals are full at the end of the algorithm. But since $|S| = \sum_{h \in G} c_h$, this implies that all of the students are also matched at the end of the algorithm, a contradiction.

To prove the stability of the final matching, consider a student s and hospital h that are not matched to each other. This can occur for two different reasons. In the first case, s never proposed to h. Since s worked her way down her preference list starting from the top, she ends up matched to a hospital she prefers to h. If s proposed to h at some point in the algorithm, it must be that h rejected s in favor of c_h students that it preferred (either at the time that s proposed, or subsequently). Since the group of students assigned to a hospital only improves over the course of the algorithm, hospital h ends up matched to c_h students it prefers to s. ■

In particular, a stable matching always exists (which is not obvious a priori!). For example, there are some simple variants of the stable matching problem for which a solution is not guaranteed.

1.4 Why Stability?

Stability is an intuitively appealing property, but is it really that important? One approach to answering this question is with theoretical justifications. For example, stability implies Pareto optimality (Exercise Set #1). It is also equivalent to the seemingly more general property that no group (of an arbitrary number of participants) can secede from the algorithm and match among themselves in a better way (with no one in the group worse off and at least one person better off).

Perhaps more compelling is the convincing empirical support for stability presented by Roth [4].[4] In the United Kingdom, back in the 1960s, residency programs decided to move from a decentralized system to centralized clearinghouses. (This was roughly 15 years after the U.S. made a similar switch.) Interestingly, the details of the implementation were left up to the individual regions, and different regions implemented different algorithms. Some

[3]Shapley shared the 2012 Nobel Prize in Economics, in part for this work. Had Gale still been alive, he surely would have also been a co-winner.

[4]Roth, a faculty member in the economics department here at Stanford, was the other co-winner of the 2012 Nobel Prize in Economics.

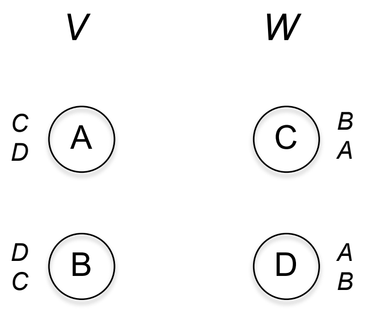

Figure 2: There can be multiple stable matchings.

of the regions implemented algorithms that guaranteed stability, while others did not. So the regions were inadvertently running a natural experiment on the importance of stability!

So what happened? Roth looked at 7 regions where detailed computer codes were available. Two of the regions copied the U.S. algorithm (i.e., deferred acceptance), and their clearinghouses remain in use today. The other five regions implemented algorithms that did not guarantee a stable matching. Three of them were quickly abandoned, after suffering from poor participation and lots of negotiations outside the system. Two remained in use. The speculation here is that these systems persist due to low-level coercion—they were the smallest regions, where everybody knew everybody, and there was a stigma around trying to circumvent the centralized clearinghouse. It is also natural to speculate that these systems suffer from participants misrepresenting their preferences and non-Pareto-optimal outcomes.

The upshot is that stability does appear to be an important property for having a functional two-sided market.

2 Student-Optimality and Strategyproofness

2.1 Multiple Stable Matchings

Next we cover your instructor's favorite property of stable matchings. First, we need to observe that the unique stable matching in the example in Figure 1 is not representative—there can be multiple (even an exponential number) of stable matchings. For example, in Figure 2, the students and the hospitals both disagree on the rankings of the others. In the matching computed by the deferred acceptance algorithm, both students get their first choice, with A and B matched to C and D, respectively. Giving the hospitals their first choices yields a different stable matching (neither hospital could be in any blocking pair).

When there are multiple stable matchings, how should we pick one? And can we say anything about the stable matching selected by the deferred acceptance algorithm?

2.2 Selecting a Stable Matching

To answer this question, let $h(s)$ denote the highest-ranked hospital (in s's preference list) to which s is matched in *any* stable matching. This is, $h(s)$ is s's best-case scenario, given that a stable matching will be implemented. The proof of the following result is in Appendix A.

Theorem 2.1 (Student-Optimality [2]) *The stable matching computed by the deferred acceptance algorithm matches every student $s \in S$ to $h(s)$.*

Theorem 2.1 implies the existence of a "student-optimal" stable matching, where every student simultaneously attains her best-case scenario. A priori, there is no reason to expect that such a matching exists.

Analogously, the matching computed by the deferred acceptance algorithm is simultaneously the worst-case scenario for all of the hospitals, with each getting the lowest-ranked students that it gets in any stable matching.

These two properties can be flipped by reversing the roles played by the students and the hospitals in the algorithm, with hospitals proposing to their favorite students, and students entertaining only their favorite offer received so far. This version of the algorithm computes a "hospital-optimal" stable matching, with every hospital h getting the best c_h students that it gets in any stable matching.

2.3 Strategyproof Considerations

Do participants in the deferred acceptance algorithm ever have an incentive to misreport their preferences? As you might have guessed, it depends on which side of the market you're on. For the usual (student-optimal) version of the deferred acceptance algorithm, honesty is always the policy for students. (While intuitive, this is surprisingly annoying to prove—see e.g. [6, Theorem 10.6.18].) Hospitals can sometimes be better off by submitting a preference list other than the honest one (this is not hard, see Exercise Set #1).

2.4 NRMP Revisited

Curiously, the original 1952 implementation of the deferred acceptance algorithm in the National Resident Matching Program was the hospital-optimal version of the algorithm. (Don't forget that the algorithm was proposed by a committee of medical school students!) The algorithm was re-implemented in the 1990s [5]. The primary motivation for the new implementation was to give couples the option of trying to get placed in geographically nearby residency programs. But in addition the algorithm was changed to favor the applicants, rather than the programs. It made a small difference but not much: empirically, each year maybe a couple percent of the students are better off than they would have been under the original algorithm. Switching from hospital-optimal to student-optimal also switches which side of the market might have opportunities for manipulating the mechanism with untruthful preference lists. But empirically, at least for the data sets arising in the NRMP, there seemed to be little incentive to be dishonest. Even assuming complete knowledge of everyone else's preference lists, less than 1% of the hospitals could have benefited by misrepresenting their preferences (or their capacity).

2.5 Key Take-Away

Here's the key take-away from this section: when choosing a stable matching in a two-sided market, you have to make a trade-off between the two sides of the market. For example, in Figure 2, you have to decide between giving the students their top choices, or giving the hospitals their top choices. But Theorem 2.1 means that: *once you've decided which side of the market to favor, no further trade-offs are necessary.* For example, if you want to favor the students, there's no need to trade off between different students—all of them agree on the best stable matching to pick.

3 Epilogue: College Admissions Revisited

Why were we talking about stable matching again? Oh right, we were thinking about college admissions, and what an alternative system might look like. The deferred acceptance algorithm gives a proposed design for a centralized clearinghouse for two-sided markets like college admissions. The NRMP shows that such clearinghouses can really work in practice, at least in some cases. Could you imagine using an analogous clearinghouse to implement college admissions in the U.S.? Is this crazy talk? Why or why not?

Before you dismiss the idea out of hand, it's worth noting that in several other countries, college admissions are indeed decided by a centralized clearinghouse. This is particularly easy in countries that have a national entrance exam. In this case, all colleges are effectively defined to have identical preferences (with students ranked according to entrance scores). This problem is easy to solve using a serial dictatorship (as in the Draw, see Lecture #1), with the students ordered according to exam score.

Hungary is an interesting case study (see [1]). There, you apply for a program (roughly equivalent to a major), not just a college. As you can imagine, different programs (e.g., technical vs. non-technical) want to give their own entrance exams, resulting in programs with different preferences. Thus in Hungary college admissions really is a full-blown two-sided market. And they essentially use the deferred acceptance algorithm to do admissions![5]

We may never see centralized college admissions in the U.S. (who would spearhead the effort)? But if we did, it would have a good chance of working well, and might reduce some of the inefficiencies and strategizing that are inevitable with the current system. Zooming out, this whole exercise was designed to fire up your imagination. Rather than accepting a long-running institution as inevitable, ask the question:

Could there be a better system?

[5]Financial aid offers are included in the stated preferences, just as room types are explicitly listed in the Draw. That is, each entry of the preference list specifies a given program at a given financial aid level.

References

[1] P. Biró. Student admissions in Hungary as Gale and Shapley envisaged. Technical Report TR-2008-291, Department of Computing Science, University of Glasgow, 2008.

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

[3] A. E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.

[4] A. E. Roth. A natural experiment in the organization of entry-level labor markets: regional markets for new physicians and surgeons in the United Kingdom. *American Economic Review*, 81(3):415–440, 1991.

[5] A. E. Roth and E. Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *American Economic Review*, 89(4):748–780, 1999.

[6] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2009.

A Proof of Theorem 2.1

As optional material, we conclude with the neat proof of Theorem 2.1.

Proof of Theorem 2.1: Let R denote the set of pairs (s, h) such that h rejected s at some point. Since each student systematically works her way down her preference list, if s is matched to h at the conclusion of the algorithm, then $(s, h') \in R$ for every h' that s prefers to h. Thus, the following claim would imply the theorem: for every $(s, h) \in R$, no stable matching pairs up s and h.

Let R_i denote the first i proposals (s, h) that get rejected. (For proposals rejected at the same time, order them arbitrarily.) We prove by induction on i that no pair in R_i is matched in any stable matching. Initially, $R_0 = \emptyset$ and there is nothing to prove. For the inductive step, suppose that the ith rejected proposal is of s by h, in favor of s'. At the time of rejection, at least one of s, s' was proposing to h.

Since s' systematically worked her way down her preference list, for every h' that s' prefers to h, $(s', h') \in R_{i-1}$. By the inductive hypothesis, no stable matching pairs up s' with a hospital she prefers to h—in every stable matching, s' is paired with h or a less preferred hospital. Since h prefers s' to s, and s' prefers h to any other hospital she might be matched to in a stable matching, there is no stable matching that pairs s with h (since s', h would form a blocking pair). ■