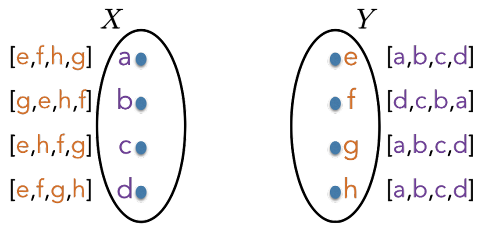
Stable Matchings

1 Stable Matching Problem

Definition (Stable matching problem)**.** An instance of the *stable matching problem* is a tuple of sets (X, Y) with $|X| = |Y|$, and a *preference list* for each element of X and Y. A preference list for an element in X is an ordering of the elements in Y, and a preference list for an element in Y is an ordering of the elements of X. Below is an example of an instance of the stable matching problem:



The output of the stable matching problem is a *stable matching*, which is a subset S of $\{(x,y) : x \in X, y \in Y\}$ with the following properties:

(i) The matching is a *perfect matching*, which means every $x \in X$ and every $y \in Y$ appear exactly once in S. If $(x, y) \in S$, we say x and y are matched.

(ii) There are no *unstable pairs*. A pair (x, y) where $x \in X$ and $y \in Y$ is called *unstable* if $(x, y) \notin S$, but they both prefer each other to the elements they are matched to.

Theorem (Gale-Shapley proposal algorithm)**.** *There is a polynomial time algorithm which, given an instance of the stable matching problem, always returns a stable matching.*

Proof. We first describe the algorithm (which is called the Gale-Shapley algorithm). For the sake of exposition, we refer to the elements of X as "companies", and the elements of Y as "students".
While there is a company x in X not matched, do the following:

- Let x be an arbitrary unmatched company.
- Let y be the highest ranked student on x's preference list to whom x has not "proposed" yet.
- Let x "propose" to y.
- If y is unmatched or y prefers x over her current partner, match x and y. (The previous partner of y, if there was any, is now unmatched.)

The theorem will follow once we show the following 3 things:

1. the number of iterations in the algorithm is at most n^2, where $n = |X| = |Y|$;
2. the algorithm always outputs a perfect matching;
3. there are no unstable pairs with respect to this matching.

Part (1) implies that the algorithm is polynomial time. Parts (2) and (3) imply that the matching returned by the algorithm is a stable matching.

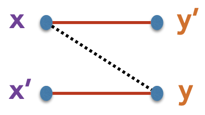
Proof of (1): Notice that the number of iterations in the algorithm is equal to the total number of proposals made. No company proposes to a student more than once, so each company makes at most n proposals. There are n companies in total, so the total number of proposals is at most n^2.

Proof of (2): The proof is by contradiction, so suppose the algorithm does not output a perfect matching. This means that some company, call it x, is not matched to any student. The proof can be broken down as follows:

$$\begin{aligned} x \text{ is not matched at the end } &\implies \text{all students must be matched at the end} \\ &\implies \text{all companies must be matched at the end.} \end{aligned}$$

This obviously leads to the desired contradiction. The second implication is simple: since there are an equal number of companies and students, the only way all the students can be matched at the end is if all the companies are matched. To show the first implication, notice that since x is not matched at the end, it got rejected by all the students it proposed to. Either it got rejected because the student preferred her current partner, or it got rejected by a student that it was already matched with. Either way, all the students that x proposed to must have been matched to a company at some point in the algorithm. But once a student is matched, she never goes back to being unmatched. So at the end of the algorithm, all the students must be matched.

Proof of (3): We first make a crucial observation. As the algorithm proceeds, a company can only go down in its preference list, and a student can only go up in her preference list. Now consider any pair (x, y) where $x \in X$, $y \in Y$, and x and y are not matched by the algorithm. We want to show that this pair is not unstable. Let y' be the student that x is matched to, and let x' be the company that y is matched to.



There are two cases to consider:

- (i) x proposed to y at some point in the algorithm,
- (ii) x never proposed to y.

If (i) happened, then y must have rejected x at some point, which implies y must prefer x' over x (recall y can only go up in her preference list). This implies y does not prefer x over her current partner, and so (x, y) is not unstable. If (ii) happened, then y' must be higher on the preference list of x than y (recall x can only go down in its preference list). This implies x does not prefer y over its current partner, and so (x, y) is not unstable. So in either case, (x, y) is stable, and we are done. □

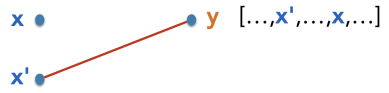
Definition (Best and worst valid partners)**.** Consider an instance of the stable matching problem. We say that $x \in X$ is a *valid partner* of $y \in Y$ (or y is a valid partner of x) if there is some stable matching in which x and y are matched. For $z \in X \cup Y$, we define the *best valid partner* of z, denoted best(z), to be the highest ranked valid partner of z. Similarly, we define the *worst valid partner* of z, denoted worst(z), to be the lowest ranked valid partner of z.

Theorem (Gale-Shapley is X-optimal)**.** *The Gale-Shapley algorithm always matches $x \in X$ with its best valid partner, i.e., it returns $\{(x, \text{best}(x)) : x \in X\}$.*

Proof. The proof is by contradiction so assume that at the end of the Gale-Shapley algorithm, there is some company not matched to its best valid partner. This means that in the algorithm, some company gets rejected by a valid partner. Consider the *first time* that this happens in the algorithm. Let x be this company and y be the valid partner that rejects x. Let x' be the company that y is matched to right after rejecting x. Note that y prefers x' over x.

During Gale-Shapley Algorithm

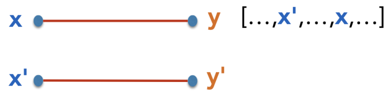
(right after x got rejected by a valid partner y)



Since y is a valid partner of x, by definition, there is some stable matching in which x and y are matched. Let y' be the match of x' in this stable matching.

Some Other Stable Matching

(where x and y are matched)



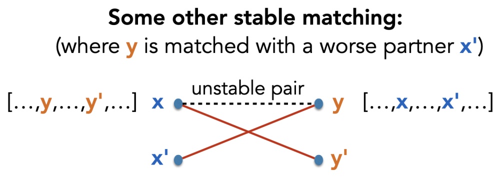
We will now show that (x', y) forms an unstable pair in the above stable matching, so in fact, the matching cannot be stable. This is the desired contradiction.

We already know that y prefers x' over x. So we just need to argue that x' prefers y over y'. And this is where we are going to use the assumption that x is the first company in the Gale-Shapley algorithm to be rejected by a valid partner. If x' actually preferred y' over y, then x' would have to be rejected by y' in the algorithm as x' later gets matched to y. This would mean x' was rejected by a valid partner before x was. Since we know this is not the case, we know that x' must prefer y over y'. This concludes the proof. □

Note (Interesting consequence of Gale-Shapley)**.** Note that it is not a priori clear at all that $\{(x, \text{best}(x)) : x \in X\}$ would be a matching, not to mention a stable matching.

Exercise (Gale-Shapley is Y-pessimal)**.** Show that the Gale-Shapley algorithm always matches $y \in Y$ with its worst valid partner, i.e., it returns $\{(\text{worst}(y), y) : y \in Y\}$.

Solution. The proof is by contradiction, so suppose that the Gale-Shapley algorithm returns a matching in which some y is matched to x, but x is not the worst valid partner of y. Let x' be the worst valid partner of y. So there is some other stable matching in which x' and y are matched. Let y' be the match of x in this stable matching.


Some other stable matching:
(where y is matched with a worse partner x')
[...,y,...,y',...]
x
unstable pair
y
[...,x,...,x',...]
x'
y'

We claim that in this stable matching (x, y) forms an unstable pair. First, y prefers x over x' because x' is the worst valid partner of y. Second, x prefers y over y' because the Gale-Shapley algorithm matches x and y, and so y must be the best valid partner of x by Theorem (Gale-Shapley is X-optimal). ■

Exercise (Is there a unique stable matching?)**.** Give a polynomial time algorithm that determines if a given instance of the stable matching problem has a unique solution or not.

Solution. Run the Gale-Shapley algorithm A with companies proposing to students. Run the algorithm A' by reversing the roles of companies and students so that students propose to companies. We claim there is a unique stable matching if and only if both A and A' output the same stable matching.

One direction is clear: if there is a unique stable matching, then A and A' must return the same matching. For the other direction, suppose A and A' return the same stable matching. We know that (i) A returns an X-optimal (company-optimal) matching (Theorem (Gale-Shapley is X-optimal)); (ii) A' returns a X-pessimal (company-pessimal) matching (Exercise (Gale-Shapley is Y-pessimal)). Since A and A' return the same matching, we must have that for all companies x, best(x) = worst(x). And this implies that in every stable matching, every x is matched to best(x) = worst(x). So there must be only one stable matching. ■

Exercise (Identical preferences)**.** Suppose we are given an instance of the stable matching problem in which all $x \in X$ have identical preference lists, and all $y \in Y$ have identical preference lists. Prove or disprove: there is only one stable matching for such an instance.

Solution. There is a unique stable matching for such an instance. To see this, let x_i be the company ranked i'th by the students, and let y_i be the student ranked i'th by the companies. Then notice that in any stable matching, x_1 and y_1 must be matched to each other because otherwise they would form an unstable pair. Given that x_1 and y_1 must be matched, x_2 must be matched to y_2 since otherwise, they would form an unstable pair. Proceeding this way, we see that for all i, x_i must be matched to y_i. ■

Exercise (Stable roommates problem)**.** Consider the following variant of the stable matching problem. The input is a set of n people, where n is even. Each person has a preference list that includes every other person. The goal is to find a stable matching. Give an example to show that a stable matching does not always exist.

Solution. Consider the example with 4 people a, b, c and d, and the following preference lists:
a: (c, b, d)
b: (a, c, d)
c: (b, a, d)

d: (a, c, b)
Notice that d is the last choice of a, b and c. On the other hand, in any stable matching, d has to be matched with someone. If d is matched with a, then (a, b) is an unstable pair. If d is matched with b, then (b, c) is an unstable pair. And if d is matched with c, then (a, c) is an unstable pair. In all cases, there is an unstable pair, so a stable matching does not exist. ■

Exercise (Soulmates)**.** Call $x \in X$ and $y \in Y$ "soulmates" if they are paired with each other in every stable matching.

1. Given $x \in X$ and $y \in Y$, design a polynomial-time algorithm to determine if they are soulmates.
2. Give a polynomial time algorithm to determine if an instance of the stable matching problem has a *unique* stable matching.

Solution.

1. Let Gale-Shapley(A, B) denote running the Gale-Shapley algorithm to match elements of A with elements in B so that the resulting matching is optimal for the elements of A. We first run Gale-Shapley(X, Y) to obtain a stable matching S_1. Next, we run Gale-Shapley(Y, X) to obtain a stable matching S_2. If x and y were matched in both S_1 and S_2, we say they're "kindred souls". We'll prove that kindred souls are soulmates, which establishes the correctness of the algorithm. We'll use the following lemma, which is a consequence of Theorem (Gale-Shapley is X-optimal) and Exercise (Gale-Shapley is Y-pessimal).

   Lemma: If $a \in A$ and $b \in B$ are paired by running Gale-Shapley(A, B), then b is a's best valid partner and a is b's worst valid partner.

   If x and y are not kindred souls, then they are not paired in S_1 or they are not paired in S_2. This means x and y aren't soulmates because there exists a stable matching where they aren't paired.

   If x and y are kindred souls, then they are paired in both S_1 and S_2. We know that y is x's best valid partner and also x's worst valid partner (using the lemma). Since y is x's best and worst valid partner, she must be its only valid partner, so x and y must be paired together in every stable matching. Therefore x and y are soulmates.

   Note that the algorithm is polynomial time since the Gale-Shapley algorithm runs in polynomial time.

2. We will use the same algorithm as in the first part and check whether S_1 is equal to S_2. They are equal if and only if there is a unique stable matching.

   The correctness can be proven as follows. By the previous part, if some pair x and y are matched in the pairing output by X-optimal (Y-pessimal) Gale-Shapley, and also matched in the pairing output by Y-optimal (X-pessimal) Gale-Shapley, x and y must be soulmates. If every couple in the matching output by X-optimal Gale-Shapley is also a couple in the matching output by Y-optimal Gale-Shapley, every couple in the matching output by Gale-Shapley are soulmates. This implies every one of these couples must be together in every stable matching. Thus, the only stable matching is the one where all these couples are together.

   Note that as before, the overall running-time is polynomial since Gale-Shapley algorithm runs in polynomial time and checking if the two matchings are the same can be done in polynomial time.

■

2 Check Your Understanding

Problem.

1. What is the definition of a stable matching?

2. Give an example of a stable matching problem instance that has at least two stable matchings.
3. What are the definitions of valid partner, best valid partner, and worst valid partner?
4. Describe the Gale-Shapley algorithm.
5. True or false: In any instance of the stable matching problem, a stable matching always exists.
6. True or false: In any instance of the stable matching problem, there is at most one stable matching.
7. What does it mean for a matching to be X-optimal?
8. What does it mean for a matching to be X-pessimal?
9. True or false: The Gale-Shapley algorithm can output different matchings based on the order of the companies proposing.
10. True or false: For every stable matching instance, the X-optimal and Y-optimal stable matchings differ in at least one pairing.
11. True or false: Given an instance of the stable matching problem, it is not possible for two companies to have the same best valid partner.
12. True or false: If the Gale-Shapley algorithm ends up pairing x with its worst possible partner y, then x and y must be paired/matched in all stable matchings.

3 High-Order Bits

Important. 1. Your main goal should be to really understand the stable matching problem, the Gale-Shapley algorithm that solves the stable matching problem, and the proof of correctness of the algorithm.

2. Best and worst valid partners are important notions. Make sure you understand what they mean and how they relate to the Gale-Shapley algorithm's output.