Game Theory
Lecture #5 – Gale-Shapley Algorithm

1 The Challenge of the Stable Marriage problem

Last time we encountered the stable marriage problem for the first time: given a set of n "men" and n "women" (where each one has a preference ranking of all the others), is it always possible to specify a matching between the two sides such that every man is paired to exactly one woman, and the matching is stable (in the formal sense defined last lecture)?

Initially, we studied the stable roommates problem and found that in that related problem, stable matchings are not guaranteed to exist. Must that negative result carry over to the stable marriage problem? As it turns out, it does not. To understand why and to answer many of our fundamental questions about the stable marriage problem, we turn now to one of the great algorithms of the 20th century.

1.1 Gale-Shapley Algorithm

Given an instance of the stable marriage problem, the Gale-Shapley algorithm works as follows. As the algorithm proceeds, it gives men opportunities to propose to women and gives women opportunities to reject proposals. Thus, the algorithm needs to keep track of which women have rejected which men. Additionally, the algorithm must track if two individuals are tentatively matched to one another. These trackers are initialized to "no men have been rejected" and "every man is single (i.e., not tentatively matched)." A round of the algorithm proceeds as follows:

- Each man who is single proposes to his top-ranked woman who has not yet rejected him.
- Any woman with at least one proposal tentatively accepts her top-ranked man who has proposed to her thus far.
- Any woman with more than one proposal rejects all but her top-ranked.
- Any man who was rejected in this round becomes single.

These rounds are repeated until a round occurs in which no man is rejected. At this point, the algorithm terminates and returns the current tentative matches. Note also that the algorithm could proceed with either men or women proposing; the difference is cosmetic.

Eventually, we need to formally address several questions:

1. Does this algorithm terminate?
2. If it terminates, does it return a matching?

3. If it returns a matching, is that matching stable?
4. If there are multiple stable matchings, which one does it return?

However, before tackling these, let us consider a few example executions of the algorithm to gain some intuition.

1.2 Example 1

Consider again the example preferences from the previous lecture:

	Ann	Beth	Cher	Dot
Al	1	1	3	2
Bob	2	2	1	3
Cal	3	3	2	1
Dan	4	4	4	4

Women's Preferences

	Ann	Beth	Cher	Dot
Al	3	4	1	2
Bob	2	3	4	1
Cal	1	2	3	4
Dan	3	4	2	1

Men's Preferences

We will denote men by $\{a, b, c, d\}$ and women by $\{A, B, C, D\}$. In the following, men are proposing and women are rejecting; to depict this, we write a column for each woman, with the list of proposals received underneath.

Round 1:
Single men at start of round : $\{a, b, c, d\}$

$$\begin{array}{cccc} A & B & C & D \\ \hline c & & a & b \\ & & & d \end{array}$$

In the first round of the algorithm, Dot receives proposals from both Bob and Dan – since Dot prefers Bob, she accepts him and rejects Dan. Now Ann, Cher, and Dot have tentative matches, and Dan is single going into round 2.

Round 2:
Single men at start of round : $\{d\}$

$$\begin{array}{cccc} A & B & C & D \\ \hline c & & a & b \\ & & d & \end{array}$$

In the second round of the algorithm, Dan proposes to Cher, his 2nd-favorite. Unfortunately for Dan, Cher prefers the mate who she is already matched with (Al), and she rejects Dan. Again, Ann, Cher, and Dot have tentative matches, and Dan is single going into round 3.

Round 3:
Single men at start of round : $\{d\}$

A	B	C	D
c		a	b
d			

At the start of this round, Dan proposes to Ann, his 3rd-ranked choice. This is not Dan's day: Ann prefers the mate who she is already matched with (Cal), so she rejects Dan. Again, Ann, Cher, and Dot have tentative matches, and Dan is single going into round 4.

Round 4:
Single men at start of round : $\{d\}$

A	B	C	D
c	d	a	b

Now Dan proposes to Beth. She is his least-favorite option, but she is the only woman left that he has not yet proposed to. Beth has no other proposals on the table, so she does not reject him. Now, no woman has more than one proposal, so no man is rejected in this round, and the algorithm terminates. The resulting matching is

Ann	Beth	Cher	Dot
|	|	|	|
Cal	Dan	Al	Bob
(3×1)	(4×4)	(3×1)	(3×1)

What does this execution of the algorithm tell us about the answers to our earlier questions? So far so good: this algorithm terminated on a stable matching. In fact, this is the same matching we found in the previous chapter when we gave each man his first choice. We lack the tools at this point to say much about the optimality of this matching, but it is worth noting that with the exception of Dan, all of the men are matched to their first choice.

1.3 Example 2

Now, as an exercise, let us examine the operation of this algorithm if we allow the women to propose to the men rather than the other way around. In the following, we present a highly compressed depiction of the operation of the algorithm.

	Ann	Beth	Cher	Dot
Al	1	1	3	2
Bob	2	2	1	3
Cal	3	3	2	1
Dan	4	4	4	4

Women's Preferences

	Ann	Beth	Cher	Dot
Al	3	4	1	2
Bob	2	3	4	1
Cal	1	2	3	4
Dan	3	4	2	1

Men's Preferences

– Round 1:

a	b	c	d
A	C	D	
B^*			

– Round 2:

a	b	c	d
A	B	D	
	C^*		

– Round 3:

a	b	c	d
A	B	C	
		D^*	

– Round 4:

a	b	c	d
D	B	C	
A^*			

– Round 5:

a	b	c	d
D	A	C	
	B^*		

– Round 6:

a	b	c	d
D	A	B	
		C^*	

– Round 7:

a	b	c	d
C	A	B	
D^*			

– Round 8:

a	b	c	d
C	D	B	
	A^*		

– Round 9:

a	b	c	d
C	D	A	
		B^*	

– Round 10:

a	b	c	d
C	D	A	B

The algorithm's output is

Ann	Beth	Cher	Dot
Cal	Dan	Al	Bob
(3×1)	(4×4)	(3×1)	(3×1)

This time, many more rounds are required, since many of the tentative matches are broken. Nonetheless, the algorithm eventually terminates at a stable matching. How can we tell this matching is stable? In fact, it is the same exact matching that resulted from our first test of the algorithm in which men proposed to women.

2 Analyzing the Gale-Shapley Algorithm

We wish to understand the Gale-Shapley Algorithm from a fundamental standpoint: what analytical guarantees can we obtain regarding the algorithm, and what do these guarantees tell us about the stable marriage problem in general?

2.1 A Termination and Correctness Guarantee

Our first two questions were "does the Gale-Shapley Algorithm always terminate?" and "if it terminates, does it return a matching?" Our first theorem answers both in the affirmative:

Theorem 2.1 *The Gale-Shapley algorithm will always terminate at a matching in a finite number of steps.*

To prove this result, we will adopt the simplifying assumption that the number of men is equal to the number of women; however, this is without loss of generality and you will address unequal numbers in the exercises.

Consider the following observations:

1. If at least one woman has no proposals, then there exists at least one woman that has multiple proposals.
2. Once a woman has a proposal, she will always have a proposal.
3. Suppose every woman has at least one proposal, then (a) every woman has exactly one proposal and (b) the Gale-Shapley algorithm terminates.

To prove 2.1, we must demonstrate that Observation 3 must eventually happen. Suppose that in some round of the algorithm, some woman (call her Ms. X) does not have a proposal. Then by Observation 1, there is some other woman (call her Ms. Y) that has at least 2 proposals (by Mr. X and Mr. Y). Since Ms. X has no proposals, Observation 2 guarantees that she has never been proposed to. In particular, Mr. X and Mr. Y have never proposed to Ms. X. In this round, Ms. Y will reject someone (say it's Mr. X) – and this person will make another proposal. If Mr. X proposes to Ms. X in the next round, we are done. Otherwise, we may repeat this argument identically for the next round.

Why does this process occur for only finitely-many steps? Note that each man can make no more than n proposals, because he cannot propose to the same woman twice. Thus, only finitely-many rounds can occur.

In fact, the worst-case number of rounds is actually linear in the algorithm's input size:

Theorem 2.2 *If there are n men and women, the Gale-Shapley algorithm terminates in at most $n^2 - n + 2$ rounds.*

Since the algorithm's input is a pair of $n \times n$ matrices (i.e., the input size is $\Theta(n^2)$), this represents a linear-time algorithm.

- Observations:
  - A man can get rejected at most $n - 1$ times
  - Every non-terminal stage there is at least one rejection
  - Every woman will receive a proposal before termination
- Proof:

  - Initial proposal: 1 stage
  - Suppose every man gets rejected exactly $n-1$ times: $n(n-1)$ stages
  - Final proposal: 1 stage
  - Total number of stages (worst-case): $1 + n(n-1) + 1 = n^2 - n + 2$

- Fact: It is impossible to have all men rejected $n-1$ times without having the Gale-Shapley algorithm terminate

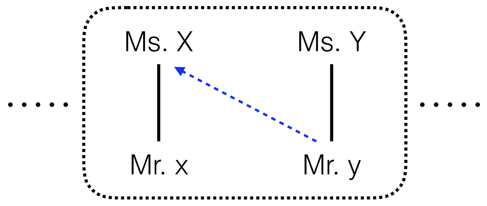
2.2 A Stability Guarantee

Having seen that the algorithm always terminates and returns a valid matching, our next question will consider the stability of that matching. The following theorem shows that indeed, the algorithm always returns a *stable* matching.

Theorem 2.3 *The Gale-Shapley Algorithm always returns a stable matching.*

To prove this theorem, we make use of the following Observation: **A woman's preference of potential match only increases by algorithm rounds.** (Why is this the case? It is simply due to the fact that women have veto power over their mates – a woman never rejects a man in favor of a less-preferred man.)

- Consider the following matching system; suppose that this matching is the output of the Gale-Shapley Algorithm.



- Suppose Mr. y prefers Ms. X over Ms. Y, and tries to get Ms. X to break off her arrangement to join him. Is it possible that Ms. X will leave Mr. x for Mr. y?
- If Mr. y prefers Ms. X to his match, then then Mr. y must have proposed to Ms. X before proposing to Ms. Y (due to the functioning of the algorithm).
- Since Mr. y proposed to Ms. Y at some point, Ms. X must have rejected Mr. y.
- At the time of rejection, Ms. X preferred some Mr. z over Mr. y.
- By our Observation above, Ms. X must prefer Mr. x over both Mr. z and Mr. y.
- Hence, Ms. X would not break off her match with Mr. x to go join Mr. y.

That is, because all of the participants in the above exercise are hypothetical, it must be the case that no pair of matched individuals prefer each other to the ones that the algorithm gave them.

Thus, the matching returned by the Gale-Shapley algorithm is stable.

2.3 Is the output optimal?

Thus far, when discussing the quality of matchings returned by the algorithm, we have avoided naming a particular notion of optimality. "Optimal" is often a slippery concept in social systems – but it turns out that the Gale-Shapley algorithm provides matchings that are optimal in a specific, strong, and yet potentially unsatisfying sense. In particular, the Gale-Shapley algorithm returns the best-possible stable matching *for each member of the proposing side.* Consider the following example preferences:

	A	B	C	D
a	3	4	1	1
b	2	2	3	4
c	4	1	2	3
d	1	3	4	2

Women's Preferences

	A	B	C	D
a	2	1	4	3
b	3	2	1	4
c	2	4	3	1
d	4	2	1	3

Men's Preferences

As it happens, this set of preference profiles has three distinct stable matchings; the first two are obtained from the Gale-Shapley algorithm (with men and women proposing, respectively). The subscripts in the following denote the preference rank of the individual's match (i.e., A_1 means that Ann is matched to her first choice):

- Matching #1: Gale-Shapley (Men proposing): $\begin{pmatrix} \mathrm{A}_3 & \mathrm{B}_3 & \mathrm{C}_3 & \mathrm{D}_3 \\ | & | & | & | \\ \mathrm{a}_2 & \mathrm{d}_2 & \mathrm{b}_1 & \mathrm{c}_1 \end{pmatrix}$
- Matching #2: Gale-Shapley (Women proposing): $\begin{pmatrix} \mathrm{A}_1 & \mathrm{B}_2 & \mathrm{C}_2 & \mathrm{D}_1 \\ | & | & | & | \\ \mathrm{d}_4 & \mathrm{b}_2 & \mathrm{c}_3 & \mathrm{a}_3 \end{pmatrix}$
- Matching #3: Alternative stable matching: $\begin{pmatrix} \mathrm{A}_3 & \mathrm{B}_2 & \mathrm{C}_2 & \mathrm{D}_2 \\ | & | & | & | \\ \mathrm{a}_2 & \mathrm{b}_2 & \mathrm{c}_3 & \mathrm{d}_3 \end{pmatrix}$

A quick scan of these matchings reveals an intriguing feature: Matching #1 seems to favor the men (i.e., all men received their 1st or 2nd choice), Matching #2 seems to favor the women (all women received their 1st or 2nd choice). In fact, *each man* prefers the match he received in Matching #1 over the match he received in either of the other two matchings, and *each woman* prefers Matching #2 in the same sense:

	A	B	C	D	a	b	c	d
Matching #1 – GS - Men	3	3	3	3	2	1	1	2
Matching #2 – GS - Women	1	2	2	1	3	2	3	4
Matching #3 – Other	3	2	2	2	2	2	3	3

That is, for this particular set of preference profiles, there is a particular stable matching that all the men can agree on, and a (different) stable matching that all the women can agree on – and the Gale-Shapley algorithm returns the proposing side's preferred stable matching.

It turns out that this feature is generic; i.e., the Gale Shapley algorithm always returns a stable matching that is optimal for each of the proposing party. In the following theorem, we call a stable matching *optimal* for a given (man, woman) if (he, she) is at least as well off under it as any other stable matching.

Theorem 2.4 *For every preference structure, the matching obtained by the Gale-Shapley algorithm, when the men propose, is optimal for each man. The matching obtained by the Gale-Shapley algorithm, when the women propose, is optimal for each woman.*

Note that this implies another remarkable fact: for any set of preference profiles, out of all possible stable matchings, there always exists a matching that all men (or all women) prefer over all other stable matchings! However, the above example shows that there need not exist a matching profile that all individuals prefer; that is, the stable matching problem may present a form of "rivalry" between the two sides. There can be cases in which the interests of the two sides are opposed to each other.

2.4 A Uniqueness Condition

We know now that the Gale-Shapley algorithm always returns a stable matching that is optimal for each member of the proposing side. However, the example in the previous section seemed to show a stable matching that existed which was *not* returned by the Gale-Shapley algorithm.

While we cannot obtain a uniqueness guarantee in general, we can use the Gale-Shapley algorithm to certify uniqueness when it is present. That is, if a stable matching is unique, the Gale-Shapley algorithm can be used to verify this fact.

Theorem 2.5 *Assuming that there is no indifference, if the matching returned by the Gale-Shapley algorithm when the men propose is identical to the matching returned by the Gale-Shapley algorithm when the women propose, then that matching is the unique stable matching.*

Recall the preference structure investigated in Examples 1 and 2 above: in these, the algorithm's outcome did not depend on which side proposed. In either case, whether the men proposed or the women proposed, the matching returned by the algorithm was the same. This is an example of Theorem 2.5 in action: by running the algorithm twice, we can definitively conclude that this preference structure has only one associated stable matching.

3 Conclusion

To conclude, let us revisit our initial application examples from the previous lecture, and consider how the Gale-Shapley algorithm can help:

Application 2.1 (Load Balancing in Distributed Computer Systems) *Suppose you operate a large video streaming platform with hundreds of servers and millions of users. At any given time, thousands of users are currently requesting to watch cat videos, and each user needs to be connected with a server so their video can be streamed. It is important to use the content delivery network efficiently (so that the servers are all mostly operating at a similar load), and it is also important to ensure low latency and quick service to users so they don't leave your platform for some other platform that treats them better. Is there a way to assign users to content servers that is fair, efficient, and keeps everyone happy?*

One way to apply the Gale-Shapley algorithm to this problem is to designate data centers as "men," and potential customer content requests as "women." Clearly, there are millions more requests than data centers (i.e., the numbers of men and women are not equal); algorithmically, this can be remedied by given each data center many slots to fill; explicitly, the matching is then performed between customer requests and service slots. The Gale-Shapley algorithm also requires preference rankings; these can be designed by letting a data center prefer geographically nearby requests over distant ones. Customer requests should prefer geographically nearby data centers which are not heavily loaded to those which are distant and under heavy loads. Note that in this circumstance, the "preferences" are themselves a design choice – so the algorithm's performance may be tuned by modifying the specifics of the preference assignments. Since content requests arrive on a near-continuous basis, the algorithm in this case will need to be run repeatedly.

Application 2.2 (Matching Drivers and Riders in a Ridesharing Platform) *On a ridesharing platform such as Uber and Lyft, millions of rides are requested per day across the world. Each time a rider requests a ride, the platform must find a driver to take that rider to her destination. To ensure a timely pickup, that driver should currently be as close as possible to that rider. Furthermore, the platform could take the rider's destination into account when selecting drivers, as some drivers may prefer not to accept trips that take them too far from home. In other words, riders have preferences over drivers, and drivers have preferences over riders. Is there a fast algorithm that can perform these matches?*

Here again, one could use the Gale-Shapley algorithm with drivers representing one side and riders representing the other. Depending on whether the platform is more driver-friendly or rider-friendly (and depending on how specifically the preferences are selected), either drivers or riders could be chosen to be the proposing side.

Note that in either example, the preferences specified in the algorithm's input need not correspond to literal human preference orderings – rather, they may represent a tool used by the system designer to shape the outcomes returned by the algorithm.

4 Questions

1. Consider the Marriage Problem discussed in lecture with the following preference structure

	Ann	Beth	Cher	Dot
Al	3	3	2	3
Bob	4	1	3	2
Cal	2	4	4	1
Dan	1	2	1	4

Women's Preferences

	Ann	Beth	Cher	Dot
Al	1	2	3	4
Bob	1	4	3	2
Cal	2	1	3	4
Dan	4	2	3	1

Men's Preferences

   (a) What is the resulting proposal associated with the Gale-Shapley algorithm with the men proposing? Is this proposal stable? Verify.

   (b) What is the resulting proposal associated with the Gale-Shapley algorithm with the women proposing? Is this proposal stable? Verify.

2. In class, we discussed how to deal with a situation where the number of men is not equal to the number of women. To handle this, simply fill out the short side of the table with "dummy" participants who are ranked last in the other side's preferences. If a person matches with a "dummy," this indicates that the person is does not have a mate. Consider the following preference structure

	A	B	C	D	E
a	1	1	2	3	3
b	2	3	1	1	2
c	3	2	3	2	1

Women's Preferences

	A	B	C	D	E
a	2	1	3	4	5
b	3	1	2	5	4
c	3	1	4	2	5

Men's Preferences

   (a) Find a stable matching system using the Gale-Shapley algorithm when the women propose.

   (b) Which women do not have a mate? Will these women remain without a mate in every other stable matching? Support your answer.

3. Consider a two-sided matching problem between a group of men {Al, Bob, Cal, Dan} and a group of women {Ann, Beth, Cher, Dot}. Suppose the matching system that resulted from the Gale-Shapley algorithm with the women proposing to the men was:

$$\begin{array}{cccc} \text{Al} & \text{Bob} & \text{Cal} & \text{Dan} \\ | & | & | & | \\ \text{Ann} & \text{Cher} & \text{Dot} & \text{Beth} \\ (3 \times 2) & (4 \times 2) & (3 \times 1) & (4 \times 2) \end{array}$$

where the numbers at the bottom corresponds to their preferences, e.g., Ann is Al's third choice and Al is Ann's second choice. Furthermore suppose the matching system that resulted from the Gale-Shapley algorithm with the men proposing to the women was:

$$\begin{array}{cccc} \text{Al} & \text{Bob} & \text{Cal} & \text{Dan} \\ | & | & | & | \\ \text{Cher} & \text{Dot} & \text{Beth} & \text{Ann} \\ (1 \times 3) & (2 \times 3) & (1 \times 4) & (3 \times 4) \end{array}$$

(a) How many stable matchings are there?

(b) Could the following matching system be stable? Justify your answer.

$$\begin{array}{cccc} \text{Al} & \text{Bob} & \text{Cal} & \text{Dan} \\ | & | & | & | \\ \text{Cher} & \text{Beth} & \text{Ann} & \text{Dot} \\ (1 \times 3) & (3 \times 3) & (2 \times 3) & (2 \times 2) \end{array}$$

Hint: You do not need to try and construct preferences.