Mechanism Design (I)
Stable Matching

Ibrahim Albluwi

Mechanism Design.

A mechanism is a procedure for making a decision or taking an action, as a function of what people want (i.e., of participants' preferences)[1]. For example, the following can be considered as *mechanism design* problems:

- Designing a procedure to pick a winning candidate in an *election* (taking into account voter preferences).
- Designing a procedure to *allocate dorm rooms* to students (taking into account student room preferences).
- Designing a procedure for an *auction* (taking into account how much participants claim they are willing to pay for the item).
- Designing a procedure for *matching* medical doctors to hospitals (taking into account which doctors each hospital prefers and which hospitals each doctor prefers).

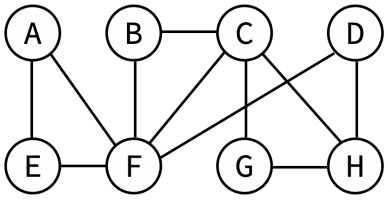
Two-Sided Markets.

A two-sided market is a market with two distinct groups of participants, each with their own preferences[1]. For examples, doctors and hospitals form a two-sided market, where each side has its own preferences regarding the other side. Other examples that where two-sided markets can be formed include: kidney donors and kidney receivers, dating websites, ride-sharing platforms, etc.

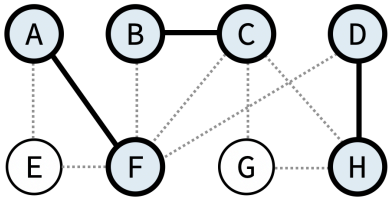
Matching.

The problem we will discuss in this lesson is a *matching* problem. Therefore, we will begin by defining some matching-related terminology.

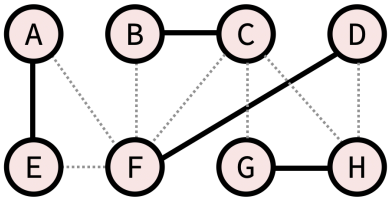
Definition. Given an undirected graph $G = (E, V)$, a *matching* is a subgraph of G where every node has degree 1. It is also called an *independent edge set*, as the subgraph is made of edges that don't have common vertices.



A graph G



Matching 1



Matching 2 (perfect)

Definition. A matching is *perfect* if all of the vertices in the graph are matched. This makes $|E|$ in the matching equal to $|V|/2$.

[1] https://timroughgarden.org/f16/l/l1.pdf

Definition. A matching is *maximal* if no more edges can be added to the matching. A matching is *maximum* if no other matching can have more edges.

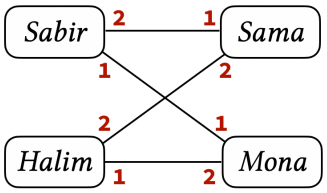
In the above example, both matchings 1 and 2 are maximal. However, matching 1 is not a maximum matching because there are other possible matchings with more edges. On the other hand, all perfect matchings are maximum matchings, because all of the vertices are matched. Note that not all maximum matchings are necessarily perfect.

The Stable Marriage Problem.

Consider a set M of men and a set W of women, where each man has a ranked list of his preferred women, and each woman has a ranked list of her preferred men. How can we match the men and women taking their preferences into account?

Consider for example the following preference lists:

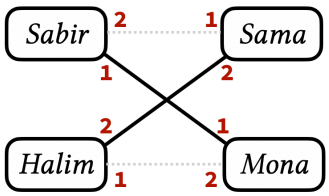
Rank	Sabir	Halim	Sama	Mona
1.	Mona	Mona	Sabir	Sabir
2.	Sama	Sama	Halim	Halim



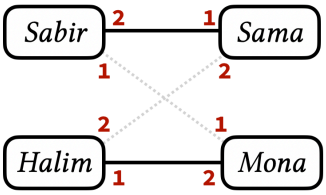
In the above example, both Sabir and Halim prefer Mona to Sama, and both Sama and Mona prefer Sabir over Halim.

There are two possible matchings:

Matching 1



Matching 2



In the Matching 1, Sabir and Mona are each other's top choice. Therefore, if Sama proposes to Sabir (because Sama does not like her match; Halim), Sabir will not accept. Similarly, if Halim proposes to Mona (because he does not like his match: Sama), Mona will not accept. Hence, we say that Matching 1 is *stable*.

Consider Matching 2 on the other hand. Mona prefers Sabir over her current match (Halim) and also Sabir prefers Mona over his current match (Sama). Hence, this matching is *not stable* and Mona and Sabir are called a *rogue couple*.

Definition. Given a matching MT, m and w are called a *rogue couple* if m prefers w over his match in MT and w prefers m over her match in MT. A matching is said to be *stable* if it does not have rogue couples.

In the Stable Marriage problem, the goal is to find a perfect matching that is stable matching, assuming the following:

- The number of men and women is the same.
- The rankings are complete (every man ranks every woman and vice versa).
- The rankings are strict (no man give the same rank to two women, and vice versa).

Applications.

There are many applications for the stable marriage problem and its variants (e.g. if the number of men and women is not the same, or if the ranking lists are not complete or strict). The following are examples:

- *Matching student groups to advisors.*
  Each advisor ranks the student groups (based on their project idea, who the students are, etc.) and each student group ranks the advisors (based on their domain of expertise, previous relationship with them, etc.).
- *Matching users to servers.*
  In large distributed systems, users prefer servers based on certain criteria (e.g. proximity) and servers prefer users based on some other criteria (e.g. service cost). Ranking lists of servers for each user and of users for each server can be computed to create an instance of the stable marriage problem.
- *Matching resident doctors to hospitals.*
  This is one of the earliest applications for the stable marriage problem (1962).

The Deferred Acceptable (DA) Algorithm (A.K.A Gale-Shapley's Algorithm).

Gale and Shapley proposed the Deferred Acceptance algorithm. We will demonstrate their algorithm using the following example.

Men

1	**2**	**3**	**4**	**5**
A	B	D	A	C
C	A	A	D	B
B	E	B	E	E
E	C	C	C	D
D	D	E	B	A

Women

A	**B**	**C**	**D**	**E**
1	2	3	1	2
3	1	5	4	3
4	5	1	2	4
2	4	4	3	1
5	3	2	5	5

To begin, we will show that a naive greedy algorithm does not produce a stable matching. If we assign for 1 his top choice, and then for 2 his top available choice, etc., we get the following matching.

$$(1, A)\ \ (2, B)\ \ (3, D)\ \ (4, E)\ \ (5, C)$$

In this matching, **4** and **D** are a rogue couple, because **4** is matched with **E** but it prefers **D** more and **D** is matched with **3** but it prefers **4** more.

Hence, a naive greedy algorithm does not always work. Next, we will describe the Deferred Acceptance algorithm.

```
WHILE there is an unmatched man and a woman he did not propose to:
    Every man proposes to the top woman on his list.
    Every woman accepts the top proposing man and rejects the other proposers.
    Every rejected man crosses out the woman who rejected him from his list.
```

The following trace using the above example shows who proposes to who in every iteration of the algorithm.

	Day 1	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**
A	**1**, 4	1	**1**, 3	1	1	1	**1**
B	2	2	2	**2**,3	2	**2**, 5	**2**
C	5	5	5	5	5,**3**	3	**3**
D	3	3,**4**	4	4	4	4	**4**
E	–	–	–	–	–	–	**5**

The following shows which women are matched (in bold) and which women are crossed out from the lists:

Men

1	2	3	4	5
A	**B**	~~D~~	~~A~~	~~C~~
C	A	~~A~~	**D**	~~B~~
B	E	~~B~~	E	**E**
E	C	**C**	C	D
D	D	E	B	A

Women

A	B	C	D	E
1	**2**	**3**	1	2
3	1	5	**4**	3
4	5	1	2	4
2	4	4	3	1
5	3	2	5	**5**

Algorithm Correctness.

To show that the algorithm is correct, we need to show that (1) it terminates, (2) it always produces a perfect matching, and (3) none of the matched couples is rogue.

Theorem 1. If there are N men and N women, The algorithm terminates in $\leq N^2$ days (iterations).

Proof.
Assume for the sake of contradiction that the algorithm does not terminate after N^2 iterations.

- If the algorithm does not terminate on a day, at least one man is rejected by a woman and crosses her out from his list (this is direct from the algorithm).
- If the algorithm does not terminate in $\leq N^2$ days, then $> N^2$ cross outs are made.
- There are N lists containing N names each (hence N^2 names in total). Therefore, at most N^2 cross outs can be made. This is a contradiction.

Lemma 1. If a woman w rejects a man m, then w accepted a man that she prefers to m.
Implication. Things get better for women or remain the same as the algorithm proceeds.

Theorem 2. The matching produced by the algorithm is perfect.

Proof.
The algorithm terminates if every man is either matched or has been rejected by all women.
Assume for the sake of contradiction that a man m was not matched by the end of the algorithm. This means that:

- m was rejected by all N women.
- Each of the N women accepted a man that she prefers to m (by Lemma 1).
- All N women are matched (to N men).
- All N men are matched. This is a contradiction.

This shows that all men are matched. A trivial counting argument can be made to show that since N men are matched, then N women must also be matched. Hence, the matching is perfect.

Theorem 3. The matching produced does not have rogue couples.

Assume for the sake of contradiction that an unmatched couple m and w is Rogue. There are only two cases for why m and w are not matched by the algorithm:

Case 1. *m* was rejected by *w*.
By Lemma 1, *w* is matched with someone she prefers to *m*. Hence the pair *m* and *w* cannot be rogue.

Case 2. *m* did not propose to *w*.
This means that *w* is lower in the ranking list of *m* than the woman that accepted *m*. Hence, the pair *m* and *w* cannot be rogue.

Fairness.

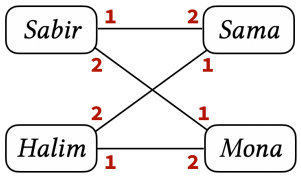
We will show next that the algorithm favors men's preferences over women's preferences. To do this, we will first define what an *feasible* partner is.

> **Definition**. Given an instance of the stable marriage problem, *m* is a *feasible* partner for *w* (or vice versa) if there exists a stable matching in which *m* and *w* are matched.

Consider the following instance of the problem.

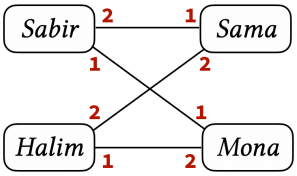
The two possible matchings for this instance are stable:

Matching **1**	Matching **2**
(*Sabir*, *Sama*)	(*Sabir*, *Mona*)
(*Halim*, *Mona*)	(*Halim*, *Sama*)



Therefore, every man is a feasible partner for every woman and vice versa.

On the other hand, consider if the rankings are changed as shown in the second graph on the right. The only stable matching in this case is Matching **2.** Hence, *Sabir* and *Sama* are not feasible partners, and *Halim* and *Mona* are not feasible partners (because there is no stable matching in which they are partners).



> **Lemma 2**. If a woman *w* rejects a man *m*, then *w* and *m* are not feasible partners. I.e. there is no stable matching in which *m* and *w* can be partners.

Proof.
Assume that Lemma 2 holds up to a certain point in the algorithm: all crossed out women are infeasible for the men who crossed them out.

Assume also that *w* has just accepted a man named *lucky*, rejected a man named *unlucky*, and that *unlucky* is about to cross out *w* from his list. We know that:

- *w* prefers *lucky* more than *unlucky* (by Lemma 1).
- *w* is the highest-ranked feasible woman on *lucky*'s list (the assumption is that Lemma 2 holds up to that point, and we know that men propose to women in decreasing preference order).

This implies that if *unlucky* is matched with *w* in some matching, then:

- *Lucky* is matched with a women he prefers less than *w*.
- *w* prefers *lucky* over her partner (*unlucky*).
- *w* and *lucky* form a rogue couple. Therefore, the matching in which *w* and *unlucky* are partners cannot be stable.

Definition. A person's *optimal* partner is the highest-ranked valid partner in his (or her) list.
A person's *pessimal* partner is the lowest-ranked valid partner in his (or her) list.

Theorem 4. The algorithm matches every man to his optimal partner and every woman to her pessimal partner.

Proof.

1. Man-Optimality. Since every man proposes to the highest-ranked woman that did not reject him, and since (by Lemma 2) all women that reject him are not feasible partners, then the first woman that accepts him is his the highest-ranked feasible candidate.

2. Woman-Pessimality. Consider the stable matching MT produced by the algorithm in which w and m are matched. Assume for the sake of contradiction that another stable matching MT_{worse} exists in which w is matched with m_{worse} that is lower on w's list than m. We know the following:

- w prefers m more than m_{worse} (by assumption).
- w is m's top feasible candidate (the algorithm is man-optimal).
- w and m form a rogue couple in MT_{worse} because each prefers the other over his/her current partner. Therefore, the stable matching MT_{worse} cannot exist.

Checking If a Stable Matching is Unique.

Given the stable matching MT, how can we check if MT is the only possible stable matching?

Assume that we run Gale-Shapley's algorithm twice: once with men being the proposers (call the result MT_1) and once with the females being the proposers (call the result MT_2). We claim that MT is unique if and only if $MT = MT_1 = MT_2$.

Proof.

(1) $\Longrightarrow$ This direction is trivial. If MT is unique, then no matter who the proposer is, the algorithm will always produce the same stable matching.

(2) $\Longleftarrow$ If $MT = MT_1 = MT_2$ then MT is both man-optimal and man-pessimal (by Theorem 4). In other words, the top feasible candidate for every man is the same as his lowest feasible candidate. This means that there is only one stable matching, because the presence of another stable matching implies that there is a match for a man that is either better or worse than his current match (i.e. the current match is either not optimal or not pessimal, which is a contradiction).

Truthfulness.

Definition. A mechanism is said to be truthful, or *strategy-proof*, if honesty is always the best policy.
In other words, if no one can gain by misreporting (lying about) their preferences.

Theorem 5. The algorithm is strategy-proof for men but not for women.

Proof.
Showing that the algorithm is strategy-proof for men is trivial knowing that it is men-optimal. Since every man m gets his top feasible candidate, then there is no way for m to improve by lying, since all the women m was not matched to are either not feasible or ranked lower in his list.

Showing that the algorithm is not strategy-proof for women can be done with an example. Consider the following instance of the problem (M = men, W = women):

M_1	M_2	M_3	W_1	W_2	W_3
W_1	W_2	W_1	M_2	M_1	M_1
W_2	W_1	W_2	M_1	M_2	M_2
W_3	W_3	W_3	M_3	M_3	M_3

Running Gale-Shapley's algorithm, we get the following matching:

$$(M_1, W_1) \quad (M_2, W_2) \quad (M_3, W_3).$$

In this matching, W_1 is matched with M_1, which is not her top choice.

Consider if W_1 misreports her preferences by saying that she likes M_3 more than M_1 (i.e. reporting the following preference list: $M_2 \succ M_3 \succ M_1$). Running Gale-Shapley, we get the following matching:

$$(M_1, W_2) \quad (M_2, W_1) \quad (M_3, W_3)$$

In this matching, W_1 is matched with M_2, which is her true top choice. I.e. she gained by being dishonest.

Can there be a different algorithm (other than Gale-Shapley's algorithm) that is strategy-proof for both men and women?

Theorem 6. No algorithm for the stable marriage problem is strategy-proof for both men and women.

Proof.
Consider the same example used above. The possible matchings produced by any algorithm are as follows:

(1)	(M_1, W_1)	(M_2, W_2)	(M_3, W_3)	⟵ *stable*
(2)	(M_1, W_1)	(M_2, W_3)	(M_3, W_2)	⟵ *not stable: M_2 and W_2 are a rogue couple*
(3)	(M_1, W_2)	(M_2, W_1)	(M_3, W_3)	⟵ *stable*
(4)	(M_1, W_2)	(M_2, W_3)	(M_3, W_1)	⟵ *not stable: M_2 and W_1 are a rogue couple*
(5)	(M_1, W_3)	(M_2, W_1)	(M_3, W_2)	⟵ *not stable: M_1 and W_2 are a rogue couple*
(6)	(M_1, W_3)	(M_2, W_2)	(M_3, W_1)	⟵ *not stable: M_1 and W_1 are a rogue couple*

Consider an algorithm that produces matching (1). If W_1 reports that she likes M_3 more than M_1 (as in the previous proof):

$$W_1: \quad M_2 \succ M_3 \succ M_1$$

The possible stable matchings become as follows:

(1)	(M_1, W_1)	(M_2, W_2)	(M_3, W_3)	⟵ *not stable: M_3 and W_1 are a rogue couple*
(2)	(M_1, W_1)	(M_2, W_3)	(M_3, W_2)	⟵ *not stable: M_2 and W_2 are a rogue couple*
(3)	(M_1, W_2)	(M_2, W_1)	(M_3, W_3)	⟵ *stable*
(4)	(M_1, W_2)	(M_2, W_3)	(M_3, W_1)	⟵ *not stable: M_2 and W_1 are a rogue couple*
(5)	(M_1, W_3)	(M_2, W_1)	(M_3, W_2)	⟵ *not stable: M_1 and W_2 are a rogue couple*
(6)	(M_1, W_3)	(M_2, W_2)	(M_3, W_1)	⟵ *not stable: M_1 and W_2 are a rogue couple*

Given that (3) is the only stable matching, this matching will be produced regardless of what the algorithm is. This shows that any algorithm that produces (1) as a matching is not strategy proof.

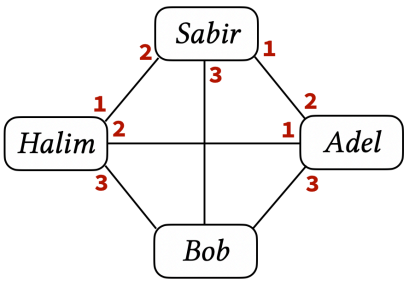
We can show using a similar argument that any algorithm that produces (3) as a matching is also not strategy-proof (M_2 can dishonestly reports his rankings to create a situation similar to the one created by W_1 above).

Given that there are only two possible stable matchings, and for each matching there is a way to gain by being dishonest, no algorithm can be strategy-proof.

Stable Roommates Problem.

Consider a list of N students (where N is even) who need to be paired into dorm rooms. If each students submits a ranking for the other $N - 1$ students, indicating who he prefers as a roommate, can we always find a stable matching?

The example on the right shows that unlike the stable marriage problem, the single-gender variant of the problem does not always have a stable matching.



In this example, the possible matchings are as follows:

Matching **1**	Matching **2**	Matching **3**
(*Halim*, *Sabir*) (*Adel*, *Bob*)	(*Halim*, *Adel*) (*Sabir*, *Bob*)	(*Halim*, *Bob*) (*Sabir*, *Adel*)
Adel - Sabir *are a rogue couple*	*Sabir - Halim* *are a rogue couple*	*Halim - Adel* *are a rogue couple*