1 Introduction([1],[2])

The problem of stable matching is defined as follows: Given a set of m resources and a group of n people, each of whose preference over the resources are represented by an order of resources, how can we assign people to resources such that the assignment has certain nice structures and properties. I have always made some generalizations here, for example, the relationship between number of resources and number of people, can preference list can be a incomplete one, is the assignment a one-to-one mapping, one-to-many, or many-to-many, or do we allow ties in preference list? These are some practical concerns for some application scenarios, but let's first ignore the generalizations and start with the most classical setting: stable marriage problem, which is a type of stable matching problem. It assumes that the number of people is the same of that of resources, preference list must be complete, one-to-to mapping, and no ties. The generalization will be covered in the extension section.

- **A little background**: The original stable matching algorithm was first introduced in 1962 by David Gale and Lloyd Shapley in the article "College Admission and the Stability of Marriage". Later on, they generalized the algorithm to deal with resident/hospital problem. Many years later, it was found that ten years earlier in 1952, National Resident Matching Program(NRMP) already started using a similar matching algorithm to match graduating medical students(residents) in the United States with hospitals. One notable difference between Gale-Shapley and NRMP matching algorithm is that the Gale-Shapley's algorithm is a resident-optimal solution while the latter favors hospital. Although Gale-Shapley's algorithm appears ten years later, the original stable marriage problem is still called Gale-Shapley's algorithm for some reason. It is also called deferred acceptance algorithm.

- **Motivation**: So why do we need stable matching algorithm? what is the motivation behind it? Imagine the situation you are a graduating medical student and you are applying for hospital internships. Assuming that each student can apply as many hospital as they want, but they can accept only one offer. Each hospital can only accept only one student. If one of the student is accepted by a relatively lower-ranked hospital and he wants to wait for more time and see if he will be accepted by some top-notched hospital. From hospital's perspective, once they extend an offer to their candidate, they want to hear back decision as soon as possible because if the decision is pending for a long time, the chances are their ideal candidate reject their offer and come to another hospital. Same thing could happen their second-ideal candidate, third, fourth... and they may end up having to accept some less-qualified people, or even no candidate at all if there is less people than hospital. To solve this issue, the internship offer is typically set with a response time. Obviously, the shorter the response time is, the happier student are because they will have more time to wait for dream hospital, the shorter the response time is, the happier the hospital is because they can find their ideal candidate as quick as possible and not run the risk of not accepting less-qualifiable people. Now we can see that It is hard to find a nice balance of response time between students and hospitals. Therefore it is necessary to design an algorithm to match student and hospital so that both of them are happy with their current assignment.

2 Gale-Shapley Algorithm([2])

Problem Statement: There are two disjoint group of man and women with size n. Each man or women has a preference list which contains all the members of the opposite sex. The preference list is strictly ordered such that there is no tie. A **matching** is a one-to-one mapping between men and women. An **instability pair** is a pair or man and women such that this women randomly find another man who ranks higher than her current partner, luckily or unluckily, the new man also found he prefers this woman than his current partner, therefore the woman and man break up with their partner and become a new pair. If there does not exist any instability pair, this matching can be called a **stable matching**. Now I will give a quick example to explain the terms I just mentioned.

Example: Consider the stable marriage instance of size 3 specified by the preference list: Man - A: (c,b,a); B: (c,a,b); C(a,b,c) and Woman - a:(B,C,A); b:(C,B,A); c:(A,B,C). This matching (Ba, Cb, Ac) is stable whereas this matching (Ab, Bc, Ca) is not stable due to the instability pair Ab.

Basic algorithm: In this section I will describe the Gale-Shapley algorithm using the traditional setting, which is man-optimal: only man has the ability to propose a marriage and then woman choose either accept or reject the proposal. Here is the idea: We first define two state for each man and woman: free and engaged. The process of Gale-Shapley algorithm can be viewed as a sequence of proposals from men to women. Each man start to propose a marriage to the woman on his preference list in the order of preference until he is engaged. If a woman is free, once she receives a proposal from a man, she become engaged immediately. If a woman is engaged and she receives proposal from another man, she will decide if she should break up with the current partner based on her preference over the two man. She then rejected the lower ranked man. If a free man's proposal is rejected, he will still be free. If he was engaged but rejected by his current partner, he will be set free by his partner and continue to propose a marriage to his next truth love on the list. This algorithm terminates when there is no free man in the list. We can see that this will definitely happen before any man exhausts his preference list due to the one-to-one mapping. The pseudocode is described as below:

Algorithm 2.1: BASIC GALE–SHAPLEY ALGORITHM(M, W)

Initially set all m in M and w in W to be free
while there is a free man m
do
- Choose the free man m
- Let w be highest-ranked woman in m's preference list to whom m hasn't proposed
- **if** w is free
  - **then** {set (m, w) become engaged
- **else if** w is engaged to m'
  - **then if** w prefer m' over m
  - **then** m is set to be free
  - **else if** w prefer m over m'
  - **then** {(m,w) become engaged; m' is set free

return (Engaged pairs)

This algorithm is a man-optimal solution because man can always propose marriage to his true love, second true love, third true love, etc... and woman will always accept the proposal from the first proposal, and can find a better man only if she receives the proposal from a higher-ranked man.You can see that using the algorithm helps man to find his best possible woman, and from woman's perspective, once she becomes engaged, she can only find a better partner later on. Therefore, this is a greedy algorithm to woman. In order to turn the man-optimal solution to woman-optimal solution, we may simply reverse the two groups.

Optimization: The basic GS algorithm can be simplified by reducing preference list for pairs that can be identified as not belonging to any stable matching, marked as impossible pair. By the impossible pairs, it means once a woman accept her first proposal from a man, she will never ever pay attention to any man ranked lower than her current partner. In this situation, the pair composed by this woman and any of these lower-ranked man is called an impossible pair. Pseudocode is given as below:

Algorithm 2.2: OPTIMIZED GALE–SHAPLEY ALGORITHM(M, W)

Initially set all m in M and w in W to be free
while there is a free man m
do
- Choose the free man m
- Let w be first woman in m's preference list
- **if** m' is engaged to w
  - **then** set m' free
- Assign m and w to be engaged
- **for** each m's successor m^* in w's preference list
  - **do** delete (m^*, w)

return (Engaged pairs)

It is interesting that, although the idea of the GS algorithm is very quite simple, it has many nice properties: 1) Using GS algorithm, we can always find at least one stable matching, 2) The man-optimal solution is also a woman-pessimal solution, likewise woman-optimal is man-pessimal, 3) The order of man's proposal does not affect the final result, and the algorithm always produce the same stable matching. In the following section we will prove the claims mathematically and analyze how these property affect the performance of the algorithm.

3 Analyze the algorithm([1],[2])

Let us go back and review how the algorithm works. In terms of the view of a woman w during the execution of the algorithm. Before any one has proposed to her, she is free. Then a man m may propose to her, and she becomes engaged. As time passes by, she may receive additional proposals, and she need to update her engaged partner based on her preference list. She accepts those men's proposals who increase the rank of her partner and break up with the lower-preference man in her preference list.([2])

Property 1. *Woman w remains engaged from the time when she receives her first proposal; and the sequence of partners to who she is engaged to gets higher and higher ranks in terms of her preference list.*

However, the case for a man m during the execution of the algorithm is quite different. He is free until he proposes to the highest-ranked woman in his preference list. At that point, he may or may not become engaged depending on the woman's availability. As time goes on, his status may alternate between free and engaged. However, the following property holds:

Property 2. *The sequence of women to whom man m proposes gets lower and lower in terms of his preference list.*

3.1 Termination

Now we can prove that the algorithm terminates, and give a bound on the maximum number of iterations needed for termination.

Theorem 1. *The $G-S$ algorithm terminates after at most n^2 iterations of the $While$ loop.*

Proof. First, we show that no man can be rejected by all the women.
Note that a woman can reject only when she is engaged, and once she is engaged she never becomes free. So the rejection of a man by the last woman on his list would imply that all woman were already engaged. But since there are equal numbers of men and women, say n, and no men has two fiancees, there can only be $n-1$ men, which is a contradiction!
Also, each iteration involves one proposal, and no man ever proposes twice to the same woman. But there are only n^2 possible pairs of men and women in total, so the total number of iterations cannot exceed n^2. Thus, termination is proved. □

3.2 Correctness: is the resulting marriage a "stable marriage"?

We prove the correctness of the algorithm by the following theorem([1]):

Theorem 2. *Consider an execution of the $G-S$ algorithm that returns a set of pairs S. The set S is a stable matching.*

Proof. Since we have equal size of men and women, S is a perfect matching. Thus, to prove S is a stable matching, we all assume that there is an blocking pair with respect to S and obtain a contradiction.
Such an block pair involves two pairs: (m, w) and (m', w'), in S with the properties that:

- m prefers w' to w.
- w' prefers m to m'.

In the execution of the algorithm that produced S, m's last proposal was, by definition, to w. Then we have the following two scenario:

- If m didn't propose to w' at some point in the execution, then w must occur higher on m's preference list than w', contradicting our assumption that m prefers w' to w.
- If m proposed to w' at some point, then he was rejected by w' in favor of some other man m'', whom w' prefers to m. m' is the final partner of w', so this implies that either $m'' = m'$ or, by Property 1, w' prefers her final partner m' to m''. If the latter one holds, them w' preference among these three men is like: $m' > m'' > m$, which is an contradiction to our assumption that w' prefers m to m'.

Thus, S is a stable matching. □

The above theorem also prove the existence of the stable matching given a full preference list from both men and women.

3.3 Classical Results

Consider the following case:

- m prefers w to w';
- m' prefers w' to w;
- w prefers to m' to m;
- w' prefers to m to m';

Any execution of $G-S$ algorithm will return the pairing: $(m, w), (m', w')$. Thus, other stable matching, consisting (m', w) and (m, w') can not be attained from any man-proposing $G-S$ algorithm. It can be reachable if we run woman-proposing $G-S$. For a larger example, with more than two people on each side, we can have an even larger collection of possible stable matchings, and many of then are not achievable by any natural algorithm.

The above example also shows a certain "unfairness" in man proposing $G-S$, favoring men. So, if the women's preference clash completely with the men's preference, then the resulting stable matching is as bad as possible for the women. So there is some destiny in which someone is fated to be unhappy: women are unhappy if men propose; men are unhappy if women propose. We may arise a natural question: Do all executions of the $G-S$ algorithm result the same matching?
This is a very basic but essential problem encountered in many settings of Computer Science: we have an algorithm that runs asynchronously, with different components performing actions that can be interacted in different ways, and we are interested in how much variability this asynchrony causes in the final outcome. For example, the independent components may not be men and women but electronic components activating parts of an airplane wing, the effect of asynchrony in their behavior can matter a lot!

However, the following classic facts about stable matching clear our doubts.

1. **Irrelevance of proposing order**: The order of proposals does not affect the stable matching produced by the man-proposing algorithm.
2. **Men-Optimality/ Woman-Pessimality**: Men-proposing GS matching S^* is man-optimal, or equivalently, it is woman-pessimal.

The stable matching generated by the man-proposing version of the $G-S$ algorithm is called man-optimal. If the roles of the sexes in the algorithm are exchanged, then this resulting woman-optimal stable matching obtaines by woman-proposing version of the $G-S$ algorithm, is optimal for the women. It may happen that the man and woman optimal stable matchings are identical, but this will not be a general case.

Now, we want to prove the above results through the following theorems. Before that, we introduce some terminology which will be used in the following theorems:

Stable partner : We say a woman w is a stable partner of man m if there is a stable matching that contains the pair (m, w).

Best stable partner : We say that w is the best stable partner of m if w is a stable partner of m and no woman, whom m ranks higher than w, is a stable partner of m.

Worst stable partner : We say that m is the worst stable partner of w if m is a stable partner of w and no man, whom w ranks lower than m, is a valid partner of w.

Theorem 3. *All possible executions of the man-proposing $G-S$ algorithm yields the same stable matching, and in this stable matching, each man has the best partner that he can have in any stable matching.*

Proof. Proof by contradiction. Suppose that some execution E of the $G-S$ algorithm results in a matching S in which some man is paired with a woman who is not his best valid partner. Note that men propose in decreasing order of his preference, this means that some man is rejected by a stable partner during the execution E. So consider the first moment during the execution E in with some man, say m, is rejected by a stable partner w. Again,

since this is the first time such rejection has occurred, it must be that w is m's best stable partner $best(m)$.

The rejection of m by w may have happened

- m proposed and was turned down in favor of $w's$ existing engagement;
- w broke her engagement to m in favor of a better proposal.

But note that, either way, at this moment, w forms or continues an engagement with a man m' whom she prefers to m.
Since w is a stable partner of m, there exists a stable matching S' including the pair (m, w). Then who is m' paired in this matching S'? Suppose it is a woman $w' \neq w$.
Here, the rejection of m by w was the first rejection of a man by a stable partner in the execution E, it must be that m' had not been rejected by any stable partner at this point in E when he became engaged to w. Since he proposed in decreasing order of preference, since w' is clearly a stable partner of m' by assumption, it must be that m' prefers w to w'. But we have already seen that w prefers m' to m, for in execution E she rejected m in favor of m'. Since $(m', w) \notin S'$, it follows that (m', w) is an instability pair in S'.
This contradict our claim that S' is stable and thus contradicts our initial assumption. □

Theorem 4. *In the man-optimal stable matching M, each woman has the worst partner that she can have in any stable matching.*

Proof. Suppose there were a pair (m, w) in M such that m is not the worst stable partner. So there is a man m', whom she likes less than m. In a stable matching S', m is paired with a woman $w' \neq w$, since w is the best stable partner of m, and w' is a stable partner of m, we see that m prefers w to w'.
But from this it follows that (m, w) is an instability in S' contradicting with the claim that S' is stable and hence contradicting out initial assumption. □

3.4 Average-case analysis([3],[4],[5])

To find the expected value of average-case proposals, we use the technique of the Principle of Deferred Decision, i.e. random choices are not all made in advance but the algorithm makes random choices as it need them.

An example of Principle of deferred decisions is clock solitaire with the first move from the pile labelled K. The game ends when you try to draw from an empty file. We win the game when all 52 cards have been drawn.
Note that at each draw any unseen card is equally likely to appear. Thus, the process of playing this game is exactly equivalent to repeatedly drawing a card uniformly at random from a deck of 52 cards. And a winning game corresponds to the last card being drawn is King.
So, the probability to win the game is $\frac{4}{52} = \frac{1}{13}$.

In order to analyze the algorithm, we make the following assumptions: assume that men's

lists are chosen independently and uniformly at random. So, rather than picking all those random preference list at the beginning, we use a random version to simulate them.
Men do not know their lists to start with. Each time, a man chooses a random woman from the women who is not already proposed by him. The preference list of man get built up during the execution. Women's lists can be arbitrary but must be fixed before the algorithm starts. Here comes a problem that a woman's choice depends on the man that proposes to her. It is difficult to analyze the average case of $G-S$ algorithm.

So we modify our algorithm as following(which is referred as Amnesiac Algorithm):
Each time, a man makes a proposal and he chooses a woman uniformly at random from the set of all n women, including those to whom he has already proposed and rejected him.

If he chooses a woman he has already proposed, then he will be rejected again since she has already rejected him, and he will try again. If not, the woman is chosen uniform at random from the women not already proposed by him. So the final pairings would be the same as $G-S$ algorithm.
The number T_A of steps for the Amnesiac Algorithm is at least as large as that of $G-S$ algorithm(since we may make repeated proposals in Amnesiac Algorithm). Or, statistically,

$$\text{for all m}, Pr(T_A > m) \geq Pr(T_{GS} > m)$$

Therefore, we do need to find an upper bound on T_{GS}, which is hard to do. We can use an upper bound on T_A to bound T_{GS}, which is easy to do. To analyze how long the algorithm takes, we need to find out how many random proposals to n women need to occur before there are no longer anyone left to propose to. This question is very similar as the coupon Collector's problem/ Occupancy problem.

- There are n types of coupons and at each trial a coupon is chosen uniformly at random. Then we want to ask: how many trials are needed to get all coupons with a reasonable probability?
- Given n distinct bins, how many balls do we need to throw such that all the bins are non-empty, with high probability?

We have already covered coupon Collector's Problem in lecture 4 and we just recap it here:

- For $0 \leq i \leq n-1$,

  $$X_i = \text{number of trials to get (i+1)th new coupon after getting i coupons}$$

  Then, $\{X_i\}$ are independent random variables,

  $$P_r[X_i = l] = p_i(1-p_i)^{l-1}, p_i = \frac{n-i}{n}$$

  X_i has geometry distribution with success probability $p_i = \frac{n-i}{n}$.
  Note that, if Y satisfies the geometry distribution with success probability p:

  $$E(Y) = \frac{1}{p}, \sigma^2(Y) = \frac{q}{p^2}$$

- $X = \sum_{i=0}^{n-1} X_i$ is the random variable representing the number of trials needed to get all coupons.
  $E(X) = E[\sum_{i=0}^{i-1} X_i = \sum_{i=0}^{n-1} E[X_i] = \sum_{i=0}^{n-1} \frac{1}{P_i} = nH_n$, where $H_n = \sum_{i=1}^{n} \frac{1}{n} = \ln n + O(1)$. Hence, $E[X] = n \ln n + O(n)$.

- $\sigma^2(X) = n^2 \sum_{i=1}^{n} \frac{1}{i^2} - nH_n$, and $\sum_{i=1}^{n} \frac{1}{i^2} \to \frac{\pi^2}{6}$ as $n \to \infty$. Here H_n is the nth harmonic function. So, according to Chebyshev inequality:

  $$Pr(X \geq n \log n + n + \epsilon n \frac{\pi}{\sqrt{6}}) \leq \frac{1}{\epsilon^2}$$

  for any ϵ.

  The above bounds is a loose bound. It states that the number of trials cannot go too large compared with the expected number.

- A stronger bound, comes from the following observation:
  Let E_i^r denotes the event that coupon i is not collected in the 1st r trials. These trials are done independently:

  $$P_r[E_i^r] = (1 - 1/n)^r \leq e^{-r/n}$$

  Let $r = E[X] = cn \log n$, then :

  $$P_r[E_i^r] = (1 - \frac{1}{n})^m \leq e^{-m/n} = e^{-c \log n} = n^{-c}$$

  The probability that we have not collected all n coupons after the expected number of trials $r = E(n) = cn \log n$ is

  $$P_r(X > E(n)) = P_r[\cup_{i=1}^{n} E_i^r] \leq \sum_{i=1}^{n} P_r[E_i^r] \leq \sum_{i=1}^{n} n^{-c} \leq n^{-c+1}$$

  Thus, with high probability, $r = E[X] = \Theta(n \log n)$.

So, back to our original problem, the expected number of proposals needed to make before there is no longer any women left to propose to is:

$$T_A = \Theta(n \log n)$$

After about $\Theta(n \log n)$ proposals under Amnesiac Algorithm, every woman will have been proposed to and the execution has a stable matching returned.

We can have an even stronger estimate for the upper bound for T_A:

Theorem 5. *For any constant c, and $m = n \ln n + cn$,*

$$P_r[T_A > m] = 1 - e^{-e^{-c}}$$

We omit the proof of the probability here since it is out of the scope of this class.

The numerical experiments also verifies the estimate of the complexity of the algorithm is $\Theta(n \log n)$ as shown in Figure(1)

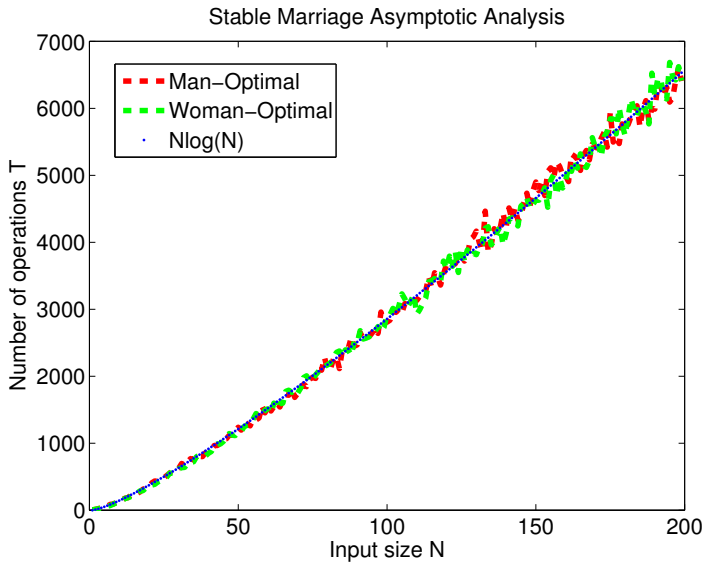

Figure 1: Asymptotic estimate of running time for $G-S$ algorithm

3.5 Running time([2])

Now we analyze the different cases of running time in stable-matching algorithm.

- Best case: the best case takes $\Theta(n)$ proposals, when the man and woman optimal stable matchings are identical.
- Worst case: the worse case takes $\Theta(n^2)$ proposals. Note that the algorithm terminates when the last woman receives her first proposal, the number of proposals cannot exceed $n(n-1)+1$, that is n proposals to each of $n-1$ women and a single proposal to the nth woman. So the complexity of the algorithm is $\Theta(n^2)$.
- Average case: the average cases of the algorithm is close related with the average number of proposals made, taken over all possible examples involving sets of size n, is $nH_n+O((\log n)^4)$, where H_n is the nth harmonic number. Here we only gives an upper bound for this average case running time. It follows that the average-case complexity of the algorithm is $\Theta(n \log n)$.

4 Extension, application and open questions

Now, we consider the following extended cases for the application of stable matching problem:

1. **Sets of unequal size([2])** In the previous section, we analyze the stable marriage problem based on the assumption that two group has the same size. However, most of the results we discussed before still apply. Now we only need modify the definition of instability. For an instability pair (m,w), they satisfy:

   (a) m and w are not partner in the stable matching M;

(b) m is either unmatched in M or prefers w to his current partner in M;

(c) w is either unmatched in M or prefers m to her current partner.

This definition of instability is based on the assumption that both woman and man wants to find a partner instead of staying single. Applying man-optimal Gale-Shapley algorithm, we will see that it terminates with each man engaged, assuming the size of man group is smaller. It gives a stable matching along with unmatched woman of size $S_w - S_m$. Similarly, we apply the woman-optimal Gale-Shapley algorithm, it also stops when each man is engaged but one difference is that the list of all the unmatched woman will be empty.

2. **Unacceptable partners([2])** Suppose that a person may declare one or more members of opposite-sex to be an unacceptable partner. In this case, the preference list of such a person contains a proper subset of the members of the opposite sex. Under these circumstances, a man and a woman can be matched only if there are acceptable to each other.
Then, we define a matching unstable if there is a man m and a woman w such that:

   (a) m and w are not partners in M, but each is acceptable to the other.

   (b) m is either unmatched in M or he prefers w to his partner in M.

   (c) w is either unmatched in M or she prefers m to her partner in M.

   Then we have the following facts regarding the $G - S$ algorithm applied to this case.

   - In a stable marriage instance that allows unacceptable partners, the men and women are each partitioned into two sets: those that have partners in all stable matchings and this that have partners in none.
   - If, in a stable marriage instance, some man m appends a previously unacceptable woman to the end of his list, then in both the man and woman-optimal stable matchings for the extended instance, no woman is worse off and no man, except possibly man m himself, is better off.

3. **Incentive Compatibility([6],[7])**. In mechanism design, a process is said to be "incentive-compatible" if all of the participants fare best when they truthfully reveal any private information asked for by the mechanism. In other words, everyone reaches their maximum benefits by telling the truth.

   Do participants have an incentive to announce a list other than their real preference lists? The answer is "yes"! In the men-proposing algorithm, sometimes women have an incentive to be dishonest about their preferences.
   But not all hope is lost! We can still show some positive thins. If in the case that when all men choose $\leq k$ women in their preference lists and do so randomly. Women give full preference lists. Then in this scenario, the number of women who would benefit from lying depends only on k and not on the number of the total people.
   First we define a Dominant-Strategy Truthful: A mechanism is dominant-strategy

truthful if it is a dominant strategy for all agents to state their truth preferences.

Moreover, holding k constants and n tends to infinity, the probability of women who have more than one stable partner tends to zero. So, even allowing women arbitrary preference lists in the probabilistic model, the expected probability of women who have more than one stable partner tends to zero.

A useful implication in economy is that:
When other players are truthful, almost surely a given player's best strategy is to tell the truth.

4. **Maximum number of Stable Matching([2])**: In some literature, it has been shown that for a stable marriage instance of size n, we can have at least 2^{n-1} different results. As we mentioned before the man-optimal solution and the woman-optimal solution is just two extreme result. For all these stable matchings, the result can be characterized as from the man-optimal to the woman-optimal solution, there is a measurement tells how happy the man is (or reversely how happy the woman is because the total happiness of man and woman will add to a constant number). All the results can be measured by this value. So we can find a lower bound of the number of stable matching for a given stable matching instance, but how about the upper bound? Is it possible to bound it? Some literature has shown that the upper bound is $n!$. That means there is actually no upper bound, since we can not bound n factorial at all. We can understand that the number of possible stable matchings is dependent on how the preference list looks like, if they extremely cross each other, then the maximum number of the stable matching is undecidable. Therefore, it remains an open question if we can bound the maximum number of stable matching.

5. **Parallel algorithm for Stable Matching([2])**: How can we efficiently run the GS algorithm for two big groups of people while still giving the stable matching in a satisfiable amount of time? How can we partition the preference in the matrix so that it will be fast for us to scan over the whole list?

4.1 Applications

- College admission;
- Resident/hospital;
- stable roommate;
- student assignment.

4.2 Other style of stable matching([8])

- Pareto-optimal matchings;
- Popular matchings;

- Rank matchings;

References

[1] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

[2] Dan Gusfield and Robert W. Irving. *The Stable marriage problem - structure and algorithms*. Foundations of computing series. MIT Press, 1989.

[3] Sariel Har-Peled. The occupancy and coupon collector problems - part ii. Lecture notes for analysis and design of algorithm, November 2005. Available from: `http://valis.cs.uiuc.edu/~sariel/teach/notes/rand_alg/lec/04_occupancy.pdf`.

[4] Sariel Har-Peled. The occupancy and coupon collector problems. Lecture notes for analysis and design of algorithm, November 2005. Available from: `http://valis.cs.uiuc.edu/~sariel/teach/notes/rand_alg/lec/04_occupancy.pdf`.

[5] William Hunt. The stable marriage problem. Lecture notes for CS684: Algorithmic Game Thoery, Jun 2004. Available from: `http://www.csee.wvu.edu/~ksmani/courses/fa01/random/lecnotes/lecture5.pdf`.

[6] Peter Cramton. Stable marriage. Lecture notes for ECON415: Market Design. Available from: `http://cramton.umd.edu/econ415/deferred-acceptance-algorithm.pdf`.

[7] Eva Tardos. Stable matching problem. Lecture notes for CS684: Algorithmic Game Thoery, May 2004. Available from: `http://www.cs.cornell.edu/courses/cs684/2004sp/may7.ps`.

[8] Kurt Mehlhorn. Remarks on matchings, Jun 2004. Available from: `http://www.mpi-inf.mpg.de/~mehlhorn/ftp/TalkCOCOA07.pdf`.