Chapter 3

The Stable Marriage Problem

3.1 Introduction

The Stable Matching Problem (SMP) introduced by Gale and Shaply [GS62] has wide applications in economics, distributed computing, resource allocation and many other fields. In the standard SMP, there are n men and n women each with their totally ordered preference list. The goal is to find a matching between men and women such that there is no instability, i.e., there is no pair of a woman and a man such that they are not married to each other but prefer each other over their partners.

We consider stable marriage instances with m men numbered $1, 2, \ldots, m$ and w women numbered $1, 2, \ldots w$. We assume that the number of women w is at least m; otherwise, the roles of men and women can be switched. The variables $mpref$ and $wpref$ specify the men preferences and the women preferences, respectively. Thus, $mpref[i][k] = j$ iff woman j is k^{th} preference for man i. Fig. 32.3 shows an instance of the stable matching problem.

In this chapter, we first give the Gale-Shapley (GS) algorithm in Section 3.3. Section 3.4 presents Algorithm α that takes any proposal vector (and therefore any matching) to a stable proposal vector in $O(m^2 + w)$ time such that the resulting proposal vector has the least distance from the initial proposal vector of all stable proposal vectors that are greater than initial proposal vector. This algorithm generalizes the GS algorithm which assumes the initial proposal vector to be the top choices. We then give a parallel LLP version of the Gale-Shapley algorithm in Section 3.5. It also discusses the constrained stable matching problem, where in addition to men's preferences and women's preferences, there may be a set of additional *lattice-linear* constraints. For example, we may state that Peter's regret [GI89] should be less than that of Paul, where the *regret* of a man in a matching is the choice number he is assigned. As another example, we may require the matching must contain some pairs called *forced pairs*, or must not contain some pairs called *forbidden pairs* [DdFdFS03].

$mpref$				$wpref$			
m_1	w_2	w_3	w_1	w_1	m_1	m_3	m_2
m_2	w_1	w_2	w_3	w_2	m_2	m_1	m_3
m_3	w_3	w_1	w_2	w_3	m_1	m_2	m_3

Figure 3.1: Stable Matching Problem with men preference list ($mpref$) and women preference list ($wpref$).

3.2 Proposal Vector Lattice

We use the notion of a *proposal vector* for our algorithms. A (man) proposal vector, G, is of dimension m, the number of men. We view any vector G as follows: $(G[i] = k)$ if man i has proposed to his k^{th} preference, i.e. the woman given by $mpref[i][k]$. If $mpref[i][k]$ equals j, then $G[i]$ equals k corresponds to man i proposing to woman j. For convenience, let $\rho(G, i)$ denote the woman $mpref[i][G[i]]$. The vector $(1, 1, \ldots, 1)$ corresponds the proposal vector in which every man has proposed to his top choice. Similarly, $(w, w, \ldots, w)$ corresponds to the vector in which every man has proposed to his last choice. Our algorithms can also handle the case when the lists are incomplete, i.e., a man prefers staying alone to being matched to some women. However, for simplicity, we assume complete lists. It is clear that the set of all proposal vectors forms a distributive lattice under the natural less than order in which the meet and join are given by the component-wise minimum and the component-wise maximum, respectively. This lattice has w^m elements.

Given any proposal vector, G, there is a unique matching defined as follows: man i and $\rho(G, i)$ are matched in G if the proposal by man i is the best for that woman in G. A man p is unmatched in G if his proposal is not the best proposal for that woman in G. A woman q is unmatched in G if she does not receive any proposal in G; otherwise, she is matched with the best proposal for her in G.

A proposal vector G represents a *man-saturating matching* iff no woman receives more than one proposal in G. Formally, G is a man-saturating matching if $\forall i, j : i \neq j : \rho(G, i) \neq \rho(G, j)$. When the number of men equals the number women, a man-saturating matching is a perfect matching (all men and women are matched). When the number of men is less than the number of women, then G is a man-saturating matching if every man is matched (but some women are unmatched). We say that a matching M_1 (or a marriage) is less than another matching M_2 if the proposal vector for M_1 is less than that of M_2. Thus, the man-optimal marriage is the *least* stable matching in the proposal lattice and woman-optimal marriage is the *greatest* stable matching.

A proposal vector G may have one or more blocking pairs. A pair of man and woman (p, q) is a *blocking pair* in G iff $\rho(G, p)$ is not q, man p prefers q to $\rho(G, p)$, and woman q prefers p to any proposal she receives in G. Observe that this definition works even when woman q is unmatched, i.e. she has not received any proposals in G. In this case, woman q prefers p to staying alone, and p prefers q to $\rho(G, p)$.

A proposal vector G is a stable marriage (or a stable proposal vector) iff it is a man-saturating matching and there are no blocking pairs in G.

3.3 Gale-Shapley Algorithm

In this section, we first present an algorithm due to Gale and Shapley for this problem (also known as the deferred-acceptance algorithm). In this algorithm, a free man proposes to women in his decreasing order of preferences, i.e., he first proposes to his top choice. Only when a man is rejected by his top choice, he would move to his next top choice. We maintain the list of all men who are not engaged in the variable $mList$. Initially all men are free (not engaged) and are in this list.

When any woman z receives a proposal from a man i, she always accepts it if she is not engaged. In this case, they both get engaged and the variable $partner[z]$ is set to i. If the woman z is engaged then she compares this proposal to her existing partner. If she prefers this proposal to her existing partner, then she breaks the engagement with her existing partner and makes i as her partner. The previous partner joins $mList$, the list of free men. If the woman z prefers her existing partner, then the proposal by man

i is rejected and the man i goes back to $mList$. Algorithm Gale-Shapley shows the pseudo-code for these steps.

Algorithm Gale-Shapley: Finding the man-optimal marriage.

```
1  input: A stable marriage instance: mpref, rank
2  output: man-optimal stable marriage
3  var
4       mList: list of 1..n; // list of men that are free initially includes all men;
5       partner: array[1..n] of 0..n initially partner[i] = 0 for all i //current fiance for woman i
6       G: array[1..n] of 0..n initially G[i] = 0 for all i //number of proposals made by man i
7  while (mList is nonempty)
8       remove a man i from mList
9       G[i] := G[i] + 1 //move to the next available top choice for man i
10      z := mpref[i][G[i]] // woman corresponding to that choice
11      if (partner[z] = 0) // the woman was not engaged
12           partner[z] := i
13      else if (rank[z][i] < rank[z][partner[z]])
14           add partner[z] to mList
15           partner[z] := i
16      else add i to get mList
17 endwhile;
18 return G;
```

It is easy to verify the following properties of the algorithm.

Lemma 3.1 *1. As the algorithm progresses, the partner for a man can only worsen and the partner for a woman can only improve.*

2. Once a woman is engaged, she stays engaged.

3. If the number of men is less than or equal to the number of women, then the algorithm is guaranteed to terminate.

3.4 Algorithm α: Upward Traversal

Algorithm α generalizes Gale-Shapley algorithm in two fundamental ways.

- *Arbitrary Initial proposal vector*: Gale-Shapley algorithm always finds the man-optimal stable marriage. Suppose that we are interested in finding a stable marriage such that the men are allowed to propose such that the proposal vector is at least I. For example, if $I = (3, 1, 2, 1)$, then the first man cannot propose to his two top choices and the third man cannot propose to his top choice. Algorithm α works even when the initial proposal vector is arbitrary instead of the top choice for each man, i.e., $I = (1, 1, \ldots, 1)$. Observe that standard Gale-Shapley algorithm does not work as is when the starting proposal vector is arbitrary. Simple Gale-Shapley algorithm would require men to make proposals and women to accept best proposals they have received so far. If the starting proposal

vector is a perfect matching but not stable, then each woman gets a unique proposal. All women would accept the only proposal received, but the resulting marriage would may not be stable.

- *Unequal number of men and women*: We consider the case when the number of women exceeds the number of men. If the number of men is greater than we can simply switch the roles in our algorithm.

  If we started with the top choices of all men, then Gale-Shapley algorithm would still return a man-optimal stable matching with the excess women unmatched. However, if we start from an arbitrary proposal vector, we can end up with all women getting unique proposals but there may exist an unmatched woman who is preferred by some man over his current match. To tackle this problem, we first do a simple check on the initial proposal vector as given by the following Lemma. Let $numw(I)$ denote the total number of unique women that have been proposed in all vectors that are less than or equal to I.

  Lemma 3.2 *Given any stable marriage instance with m men and the initial proposal vector I there is no stable marriage for any proposal vector $G \geq I$ whenever $numw(I)$ exceeds m.*

  Proof: Consider any proposal vector $G \geq I$. Since the total number of men is m, there is at least one woman q who has been proposed in the past of G who does not have any proposal in G. Suppose that proposal was made by man p. Then, man p prefers q to $\rho(G, p)$ and q prefers p to staying alone.

  ■

  Hence, in our algorithm we only consider I such that the total number of women proposed until I is at most m.

Even when the number of men and women are equal, the proposal vector may be a perfect matching but not stable. To address this problem, we first define $forbidden(G, i)$ as the predicate that there exists another man j such that either (1) both i and j have proposed to the same woman in G and that woman prefers j, or (2) $(j, \rho(G, i))$ is a blocking pair in G.

We first show that

Lemma 3.3 *Let G be any proposal vector such that $numw(G) \leq m$. There exists a man i such that $forbidden(G, i)$ iff G is not a stable marriage.*

Proof: First suppose that there exists i such that $forbidden(G, i)$. This mean that there exists a man j such that j has proposed to the same woman and that woman prefers j or $(j, \rho(G, i))$ is a blocking pair in G. If both i and j have proposed to the same woman in G, then it is clearly not a matching. If $(j, \rho(G, i))$ is a blocking pair then G is not stable.

Conversely, assume that G is not a stable marriage. This means that either G is not a perfect matching or there is a blocking pair for G. If it is not a perfect matching, then there must be some woman who has been proposed by multiple men. Any man i who is not the most favored in the set of proposals satisfies $forbidden(G, i)$. If G is a perfect matching but not a stable marriage, then there must be a blocking pair (p, q). If q has been proposed in G by man i, then $(p, \rho(G, i))$ is a blocking pair. If q has not been proposed in G then we know at least $m + 1$ women that are in $numw(G)$ which violates our requirement on G.

■

Algorithm UpwardTraversal exploits the $forbidden(G, i)$ function to search for the stable marriage in the proposal lattice. The basic idea is that if a man i is forbidden in the current proposal vector G, then he must go down his preference list until he finds a woman who is either unmatched or prefers him to her current match. The while loop at line (1) iterates until none of the man is forbidden in G. If the while loop terminates then G is a stable marriage on account of Lemma 3.3. At line (2), man i advances on his preference list until his proposal is the most preferred proposal to the woman among all proposals that are made to her in any proposal vector less than or equal to G. If there is no such proposal, then there does not exist any $G \geq I$ such that G is stable and in line (3), the algorithm returns null. Otherwise, the man makes that proposal at line (4).

Algorithm UpwardTraversal: An Algorithm that returns the man-optimal marriage greater than or equal to the given proposal vector I.

```
1  input: A stable marriage instance, initial proposal vector I
2  output: smallest stable marriage greater than or equal to I (if one exists)
3  forbidden(G, i) holds if there exists another man j such that either (1) both i and j have proposed
     to the same woman in G and that woman prefers j, or (2) (j, ρ(G, i)) is a blocking pair for G.
4  if numw(I) > m then return null; // no stable matching exists
5        else G := I;
6  while there exists a man i such that forbidden(G, i)
7        find the next woman q in the list of man i s.t. i has the most preferred proposal to q until G,
8        if no such choice after G[i] or the number of women proposed including q exceeds m then
9              return null; // "no stable matching exists"
10       else G[i] := choice that corresponds to woman q;
11 endwhile;
12 return G;
```

3.5 An LLP Algorithm for the Stable Matching Problem

We now derive the algorithm for the stable matching problem using Lattice-Linear Predicates. We let $G[i]$ be the choice number that man i has proposed to. Initially, $G[i]$ is 1 for all men. For convenience, let $\rho(G, i)$ denote the woman $mpref[i][G[i]]$.

Definition 3.4 *An assignment G is feasible for the stable marriage problem if (1) it corresponds to a perfect matching (all men are paired with different women) and (2) it has no blocking pairs.*

We show that the predicate "G is a stable marriage" is a lattice-linear predicate.

Lemma 3.5 *The predicate that a vector G corresponds to a stable marriage is lattice-linear.*

Proof: Let z be $\rho(G, j)$, the woman that corresponds to choice $G[j]$ for man j. We define j to be forbidden in G if there exists a man i such that z prefers man i to man j and either man i has also been assigned z in G or he prefers z to his current choice, i.e., man i and woman z would form a blocking pair in G. Formally, $forbidden(G, j)$ is defined as $(\exists i : \exists k \leq G[i] : (z = mpref[i][k]) \wedge (rank[z][i] < rank[z][j]))$.

It is easy to see that G is not a stable marriage iff $\exists j : forbidden(G, j)$. If G is not a perfect matching then there must be at least one woman who is assigned to two men. In that case, the less preferred man is forbidden. If G is a perfect matching but has a blocking pair, then the partner of the woman in the blocking pair is forbidden. Conversely, $forbidden(G, j)$ implies that either G is not a perfect matching or has a blocking pair.

We only need to show that if $forbidden(G, j)$ holds, then there is no proposal vector H such that $(H \geq G)$ and $(G[j] = H[j])$ and H is a stable marriage.

Consider any H such that $(H \geq G)$ and $(G[j] = H[j])$. We show that H is not a stable marriage. Since $G[j]$ is equal to $H[j]$, $\rho(G, j)$ is equal to $\rho(H, j)$. Let i be such that $\exists k \leq G[i] : (z = mpref[i][k]) \wedge (rank[z][i] < rank[z][j]))$. Since $G \leq H$, $G[i] \leq H[i]$, we get that $\exists k \leq H[i] : (z = mpref[i][k]) \wedge (rank[z][i] < rank[z][j]))$. Hence, $forbidden(H, i)$ also holds.

■

Lemma 3.5 immediately gives us the Algorithm LLP-ManOptimalStableMarriage. The **always** section defines variables which are derived from G. These variables can be viewed as macros. For example, for any thread $z = mpref[j][G[j]]$. This means that whenever $G[j]$ changes, so does z.

If man j is forbidden, it is clear that any vector in which man j is matched with z and man i is matched with his current or a worse choice can never be a stable marriage. Thus, it is safe for man j to advance to the next choice.

Algorithm LLP-ManOptimalStableMarriage: A Parallel Algorithm for Stable Matching

```
1 P_j: Code for thread j
2 input: mpref[i,k]: int for all i,k; rank[k][i]: int for all k,i;
3 init: G[j] := 1; // works for any initialization
4 always: z = mpref[j][G[j]];
5 forbidden: (∃i : ∃k ≤ G[i] : (z = mpref[i][k]) ∧ (rank[z][i] < rank[z][j]))
6       advance: G[j] := G[j] + 1;
```

We now present an algorithm to find stable marriages that satisfy additional constraints. Due to Lemma 2.6, we can use LLP algorithm to find the least stable marriage satisfying these constraints. The following lemma proves lattice-linearity of many such constraints.

Lemma 3.6 *The following constraints are lattice-linear.*

1. *The regret of man i is at most that of the regret of man j.*
2. *Man i cannot be married to woman j.*
3. *The regret of man i is equal to that of man j.*

Proof: Let B be the predicate that G is a stable marriage and it satisfies the corresponding additional constraint.

1. Suppose that G is a stable marriage but it does not satisfy B. This means that regret of man i is more than the regret of man j. In this case, we have $forbidden(G, j)$, because unless $G[j]$ is advanced, the predicate cannot become true.

2. If $\rho(G, i) = j$, then $forbidden(G, i)$ holds.
3. This condition is a conjunction of two lattice-linear conditions of type in part (1).

■

We now enumerate advantages of the LLP algorithm.

1. It gives us a more general algorithm. Instead of starting from the initial vector, we can start from any vector and find a stable marriage greater than or equal to that vector, if one exists.
2. We can use the LLP algorithm for finding a stable marriage that satisfies additional constraint such as the regret of man i is at most that of the regret of man j.
3. We can automatically deduce that the intersection of two stable marriages is also a stable marriage.
4. The LLP algorithm is useful in the context of parallel computing. If two men are forbidden, then both of them can advance in parallel.
5. The LLP algorithm is also useful in the context of distributed computing. The man i may not have the most recent information about man j. Even then, the algorithm will terminate with the stable marriage that is the least in the proposal vector lattice.

3.6 Summary

The following table lists all the algorithms discussed in this chapter.

Problem	**Algorithm**	**Parallel Time**	**Work**
Stable Marriage	Gale-Shapley	$O(n^2)$	$O(n^2)$
Stable Marriage	α	critical path length	$O(m^2)$
Stable Marriage	LLP	critical path length	$O(n^2)$

3.7 Problems

1. Show that the Gale-Shapley algorithm always finds a stable marriage.
2. Give an instance of the stable marriage problem in which there is only one stable marriage: the last choice for all men.
3. Give an instance of the stable marriage problem in which the maximally parallel LLP version of the algorithm is n times faster than the sequential Gale-Shapley algorithm.
4. Show that the set of stable marriages is closed not only under meet, but also the join operation. Thus, the set of stable marriages form a sublattice of the proposal lattice.
5. Give an algorithm in which all men start with their least favored choice and improve their choices in the search of the stable marriage.

3.8 Bibliographic Remarks

The stable matching problem has been studied extensively with multiple books and survey articles devoted on the topic [GI89, Knu97, RS92, IM08].