FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	V čase proměnlivá stabilní párování
Student:	Bc. Dominik Šmejkal
Vedoucí:	doc. RNDr. Dušan Knop, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Stable Marriage je fundamentální problém teorie stabilních párování. Vstupem problému jsou (stejně velké) množiny agentu M a F. Každý z agentů má preference vyjádřené jako (totální) lineární uspořádání na agentech z opačné množiny. Cílem je najít párování mezi agenty, kde nebude existovat dvojice m\in M a f\in F taková, že m preferuje f před aktuálně přiřazeným párem a f preferuje m před aktuálním partnerem. Pro tento problém existuje slavný polynomiální algoritmus, se kterým přišli v roce 1962 Gale a Shapley. Shapley za výzkum v oblasti stabilních párování dokonce získal Nobelovu cenu.

Cílem práce je studium problému z hlediska temporálních preferencí, které dobře zachycují dynamicky se měnící svět. Práce se bude věnovat zejména hledání temporálních interpretacích tohoto problému, se speciálním zaměřením na algoritmické a těžkostní výsledky.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 28. dubna 2022 v Praze.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Temporal Stable Matchings

Bc. Dominik Šmejkal

Department of Applied Mathematics
Supervisor: doc. RNDr. Dušan Knop, Ph.D.

May 4, 2023

Acknowledgements

V první řadě bych chtěl poděkovat svému školiteli, Ing. Šimonovi Schierreichovi, za jeho perfektní vedení mého průběhu psaní diplomové práce. Také bych rád poděkoval rodinně a přítelkyni za jejich podporu a pochopení v době psaní práce.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 4, 2023 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Šmejkal, Dominik. *Temporal Stable Matchings*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstract

In dynamically changing world, we cannot expect our preferences to be constantly unchanged under outside factors. Hence, we need solutions of our problems to adapt well to changes. In Stable Marriage problem, sets of men and women are being matched according to their *preferences*. The goal is to find a matching, where no man and woman both prefer each other over their current partners. We study the effect of dynamicaly changing preferences of these men and women and define Temporal Stable Marriage problem that adresses these changes. This problem tries to find a matching, where at most k pairs prefer each other over their current partners under multiple preference profiles. We prove that this problem is *NP-complete*, even with constant value of k, and propose algorithms that might be a few observations away from being able to find a valid solution in time better than naive brute force method.

Keywords Stable Marriage, NP-complete, Polynomial Reduction, Dynamic programming, Blocking pair, Temporal graph, Bipartithe graph

Abstrakt

V dynamicky se měnícím světě nemůžeme očekávat, že naše preference zůstanou neměnné pod vlivem vnějších faktorů. Proto potřebujeme, aby se řešení našich problémů dobře adaptovaly těmto změnám. V problému Stable Marriage se skupiny mužů a žen párují podle jejich *preferencí*. Cílem je najít takové párování, kde neexistuje muž a žena, kteří se vzájemně preferují před svými aktuálními partnery. Efekt dynamicky se měnících preferencí těchto žen a mužů studujeme v rámci naší definice problému jako Temporal Stable Marriage, který tyto změny řeší. Řešením tohoto problému je nalézt párování, ve kterém se nejvýše k párů preferuje vzájemně před jejich přiřazenými partnery pod více preferenčními profily. Dokazujeme, že tento problém je *NP-úplný*, i s konstantní hodnotou k, a navrhujeme algoritmy, které jsou potencionálně už jen pár pozorování vzdálené od schopnosti vracet správné řešení v čase lepším, než naivní algoritmus hrubou silou.

Klíčová slova Stabilní párování, NP-úplný, Polynomiální redukce, Dynamické programování, Blokující hrana, V čase proměnlivý graf, Bipartitní graf

Contents

List of Figures

Introduction

The need to be matching agents in pairs is present in a diverse range of fields and industries. Ranging from mobility as a service and food delivery applications, where we need to match the driver and the customer, to organ donation, where we need to match the donor and the recipient, to web advertisements, where it is needed to match the advertisement and the advertisement space, fast and, ideally, optimal matching is needed everyday.

One of the options for a model of optimal matching is *stable matching* in Stable Marriage problem. It is a classic problem in the field of mathematics and computer science that deals with matching pairs from two different sets, often labeled as *men* and *women*. Each one of these men and women have their *strict preferences* for their perspective partners, effectively ordering them from the most preferred partner to the least preferred partner. The question of this problem is: *Does a matching, where no men and women prefer each other to their respective partners, exist?*

The problem was first formally defined by *David Gale* (December 13, 1921 – March 7, 2008) and *Lloyd Shapley*[1] (June 2, 1923 – March 12, 2016) in 1962, and they showed that the required matching, which is called *stable matching*, always exist. It's proof was constructive, using the very important algorithm known as Gale-Shapley Algorithm or Deferred Acceptance Algorithm.

In this thesis, we study how to utilize stable marriage problem in a changing environment, where the preferences of men and women may be changing. We are going to define a temporal interpretation of Stable Marriage problem and examine it's properties.

[1] 50 years after David Gale and Lloyd Shapley first introduced the Stable Marriage problem, in 2012, Shapley and Alvin E. Roth (who made significant contributions to practical applications of Stable Marriage) received the Nobel Memorial Prize for "*the theory of stable allocations and the practice of market design*". The reason why David Gale was not awarded the Nobel Memorial Prize is that it is not awarded posthumously.

Thesis Structure

At first, we are going to introduce the Stable marriage problem, explore it's applications and research which modifications of this problem were already analyzed. We are going to show some of the complexity and algorithmic results of these modified versions of Stable Marriage. We are also going to talk about problems similar to Stable Marriage problem.

After we establish the current state of research, we are going to formally define Stable Marriage, standard definitions and lemmas surrounding it. As different authors tend to use slightly different notation, we are going to unify the theory from different sources.

After showing the Gale-Shapley Algorithm, we are going to focus on selected literature, that is expected to help us with tackling our Temporal Stable Marriage problem later. These are going to be examined in detail. We are also going to show some modified versions of Stable Marriage problem more in detail, when they offer interesting solutions.

Finally we are going to introduce our definition of Temporal Stable Marriage problem and show some of it's properties. In the end, we are going to discuss some of our unsuccessful approaches to face our problem, and the reasons why we suppose they are not working as first expected.

Goals

Main goals of this thesis are to study the Stable Marriage problem from the viewpoint of dynamically changing world. The thesis will focus mainly on finding temporal interpretations of this problem, with a special emphasis on algorithmic and complexity results.

We will define Temporal Stable Marriage problem and analyze it's properties, mainly focusing on:

- proving, that the problem is *NP-complete* (resp. *NP-hard*) by providing a polynomial reduction from another *NP-complete* problem,
- proving, that it belongs to P, by finding an algorithm solving Temporal Stable Marriage in polynomial time,
- finding algorithms for special cases of Temporal Stable Marriage problem.

Chapter 1

Stable marriage

We are going to formally define Stable Marriage instance and it's preference list first. Later, additional definitions like stability will be introduced. Definitions, lemmas and theorems in this chapter are based mostly on papers [1, 2, 3, 4, 5, 6, 7], unless it is said otherwise. As each publication uses a slightly different definitions and notation, we are going to use unified definitions and notation in the whole thesis, therefore they may differ from the papers cited.

First, we are going to look at a summary of related work on Stable Marriage problem in Section 1.1. After that we show essential definitions for Stable Marriage problem, and Gale-Shapley Algorithm later.

1.1 Stable Marriage Problem Research Overview

The Stable Marriage problem is a classic problem in mathematics and computer science that deals with finding a *stable matching* Q between sets of men and women based on their preferences for each other. Specifically, given n men and r women, each with their own strict preference relation ordering members of the opposite sex, the problem is to find a matching where no two agents would both prefer to be matched with each other over their current partners. If such pair exists for this matching, it is called *blocking pair*, else we have found stable matching.

We can find a stable matching using Gale-Shapley Algorithm in $O(nr)$ time. This algorithm is also used as a proof for the existence of stable matching in any instance of the Stable Marriage problem.

The solution to this problem has important applications in various fields, including economics, game theory, and network theory. Some of these applications are:

- **Job Matching/School Choice:** Companies or schools that are receiving applications can use the Stable Marriage problem to match

applicants with the positions/capacities that best fit their qualifications and preferences. Examples of this application are *National Resident Matching Program* [8], *Canadian Resident Matching service* [9] and *Japan Residency Matching Program* [10], where medical graduates are matching with country's hospitals and their available positions. This is usually modeled as a *Hospitals-Residents* or *College admissions* problem, which is a slightly modified version of the Stable Marriage problem, where a single hospital/company/school can accept multiple residents/applicants/students. This problem was solved in the same paper [1] by Gale and Shapley where the Stable Marriage problem was solved.

- **Organ Transplants:** The problem of finding a compatible organ donor for a patient in need of a transplant can be modeled as a Stable Marriage problem. In this case, the patients represent one set of individuals, and the potential donors represent the other set. Kidney exchange application model is mentioned often in multiple papers [11, 12, 13, 14, 15] and studied extensively (especially by Alvin E. Roth), but we have not been able to find any proof of an actual kidney exchange using this algorithm. Most sources tell us that the exchange currently happens either via a swap of partners in two donor and recipient pairs, or by NEAD – Never Ending Altruistic Donor chain. NEAD starts with one unmatched donor, who pairs with a compatible recipient waiting for transplant and the recipient's incompatible donor, that was paired to him earlier is paired with another compatible recipient, creating an infinite chain. [16]
- **Stable Allocations in Content Delivery:** Because of the need of content delivery networks to balance the load to the servers, Stable Marriage is used to match clients (map units) and clusters of Akamai content delivery network, according to Bruce M. Maggs and Ramesh K. Sitaraman [17]. It uses a generalized definition of Stable Marriage problem, utilizing:
  - partial preference lists, where the preferences do not need to be defined for every agent in the opposite set, as there are thousands of clusters and tens of milions of map units, and
  - many-to-many matchings, where the demands of map units and capacities of clusters are possibly higher than one.

1.1.1 Modified versions of problem

In this section we look at the most important modifications of Stable Marriage problem that were already analyzed. In addition to the standard definition above, there exists a number of it's modifications. One of them was

already mentioned – *Hospitals-Residents* or *College admissions* problem. It adds an option for one set of agents (the hospital/college set) to have a higher capacity. These agents then can be in a pair with multiple partners at the same time, up to their capacity. This is an abstraction for the fact that for example colleges have many places for new students which are all equal.

Incidental to this definition is an important theorem formulated by Alvin E. Roth [18] and David Gale and Marilda Sotomayor [19]. Nowadays called *Rural Hospitals Theorem*, it proves that every matching, that is stable, is made by the same subsets of men and women. This means that an agent who is not part of some stable matching is left out in every stable matching. Additionally, for the *Hospitals-Residents* problem it also means that any hospital that fails to fill it's capacity in some stable matching will not only fill the same number of positions in every other stable matching, but will also match the same set of residents. It's importance lies in the fact that the selection of subsets of agents is not a property of the given stable matching, but it is a property of the given matching instance. [20]

1.1.1.1 Incomplete Preferences and Ties

Another modification is a definition that allows men and women to have incomplete preference lists, where partners who are not on preference list are not considered acceptable for the owner of the list. This version is called Stable Marriage with Incomplete preference lists, often shortened as SMI. [7]

Another common modification of the basic definition is Stable Marriage with Ties, shortened as SMT, adds the option to rank the members of the opposite sex in non-strict order. Because of the ties incorporated in preferences, the definition of stability for this problem was extended to *super-stability*, *strong stability*, and *weak stability*.

Super stability redefines blocking pairs as not strictly preferred pair to the current matching for both agents. Strong stability requires the blocking pair to contain at least one agent that prefers this pair to his current match. Weak stability has the strongest requirements on a blocking pair, as it requires both agents in the blocking pair to prefer each other strictly to their current matches.

It is easy to see that a super-stable matching is strongly stable and strongly stable matching is weakly stable. Unlike weakly stable matching, the other two may not exist in a given instance of the Stable Marriage problem. Weakly stable matching can be found in $O(nr)$ time. The existence of super-stable matching is verified by finding one in $O(nr)$ time, and for strongly stable matching this takes $O(l^4)$ (where $l = \max(n, r)$) with algorithm strong by Irving [21], resp. $O(l^3)$ with algorithm by Kavitha [22].

A combination of the last two modifications, Stable Marriage with Ties and Incomplete preference lists (SMTI), allows both of these

modifications at the same time. This problem has the same extended definitions of stability and the same time complexities for verification of existence of super-stable and strongly stable matchings. Additionally, both of these types of stability have the same size of all stable matchings for an instance of Stable Marriage with Ties and Incomplete preference lists. [7]

Weakly stable matching exists in every instance of Stable Marriage with Ties and Incomplete preference lists and can be found in $O(nr)$ time. But now, one instance may have stable matchings of different sizes. A problem of finding the larges one, known as *MAX SMTI* problem, was shown to be *NP-complete* by David F. Manlove [4] and Iwama [23].

1.1.1.2 Robustness

A question of *robustness* of stable matching was studied mostly by B. Genc, M. Siala, G. Simonin, and B. O'Sullivan [24, 25] since year 2017. By their definition, the *robustness* of a stable matching is measured by the number of modifications required to find a different stable matching.

They introduced the notion of (a, b)-supermatch. They define it as a stable matching Q, where we can break up at most a pairs and find a different stable matching by changing the partners of agents from those a pairs and at most b other pairs.

In this context, $(1, b)$-supermatch where b is minimal, is considered the most robust matching. It is shown that checking whether a given stable matching is $(1, b)$-supermatch can be done in polynomial time. Genetic algorithm, local search algorithm and constraint programming model to find the most robust $(1, b)$-supermatch is also shown. [25]

It was also shown that the problem of deciding if there exists a $(1, 1)$-supermatch or $(1, b)$-supermatch is *NP-complete*. About (a, b)-supermatch we only know that it is *NP-hard* and we also know that $(2, 0)$-supermatches do not exist. [26, 27]

1.1.2 Similar problems

In this section we explore other similar problems. We chose Stable roommates and Popular Marriage as the representatives.

1.1.2.1 Stable Roommates

The most well known similar problem to Stable Marriage is probably the Stable Roommates problem, first presented by Knuth [28]. The way it differs from Stable Marriage the most is that there is only one set of agents, who try to create a stable matching among themselves. Every agent has a preference relation ordering others strictly in order of preference. Blocking pairs work the same way as with Stable Marriage.

The main difference of the two problems is that stable roommates instance may not have a stable matching. It is easy to see, that for $n = 3$, when every agent prefers the agent who does not prefer them to the other option, no matching can be stable, as the third agent is always going to block the other two.

Still, polynomial algorithm for finding a stable matching, if it exists, was found by Irving [29]. This algorithm has $O(n^2)$ time complexity and two phases.

In the first phase, every agent starts making proposals to other agents in order of preference. If agent receives multiple proposals, he keeps the most preferred and rejects the others. At the end of this phase, every agent is proposing to someone and is proposed to by some other agent. This reduces the sizes of preference lists of agents.

The second phase is about eliminating rotations (different rotations than our rotations defined in Definition 11). This rotation is a cyclic sequence of distinct agents, where the second currently most preferred to one agent is the first most preferred to the next agent in the sequence. This *"all-or-nothing cycle"* is used for iteratively eliminating pairs (for details refer to Irving [29]) from the preference lists of agents, until the size of preference lists for all agents is equal to one. If it becomes empty, that means that the instance of stable roommates admits no stable matching.

1.1.2.2 Popular Matchings in Stable Marriage

Stable matching exists in every instance of the Stable Marriage with Incomplete Preference Lists problem, thought there may exist a matching with higher cardinality than the stable matchings. Péter Biró, Robert W. Irving and David F. Manlove [30] propsed the use of *popular matchings* in Stable Marriage problem (and also Stable Roommates problem). They define that one matching is more popular than the other, if more agents prefer their matches. Matching is considered as popular, if no other matching is more popular. They also define a strongly popular matching as a matching that is strictly more popular than any other matching (it's existence can be verified in $O(nr)$ time [30]).

Chien-Chung Huang and Telikepalli Kavitha [31] have shown that a stable matching is actually a popular matching with minimal cardinality. They have also proposed an algorithm for finding the maximum cardinality popular matching in $O(nrk)$ time, where $k = min(n, r)$, and extended the theory surrounding this algorithm in [32], where they have shown that the maximal cardinality popular matching can be found even when $k = 2$, hence $O(nr)$ time.

This algorithm is a simple modification for Gale-Shapley Algorithm, where the men that already proposed to every woman in their preference list start proposing to them again from the start. But now, they gain one

"wildcard", that makes them more preferred when proposing than the agents with less *wildcards*. It is shown, that returning agents two times is enough for the algorithm to return the maximal cardinality stable marriage. [32]

1.2 Basic definitions and theorem

In this section we formally define Stable Marriage problem and other essential definitions. We also show a major theorem about the existence of *stable matching*.

Definition 1 (Stable marriage)**.** An instance $I = (M, W, \succ)$ of Stable Marriage consists of n men $M = \{m_1, \ldots, m_n\}$ and r women $W = \{w_1, \ldots, w_r\}$ and preference relations $\succ = (\succ_i)_{i \in M \cup W}$, where each person has a preference relation that strictly orders members of the opposite sex in strict order of preference.

Note 1. Even though the sizes of sets of men and women can be different, we are going to assume $n = r$, as we can add agents to the smaller set with arbitrary preference relations to make the sizes equal. For the agents from the formerly bigger set, these new agents are the least preferred, in arbitrary order.

Definition 2 (List of Preferences)**.** Let $I = (M, W, \succ)$ be an instance of the Stable Marriage problem. Function pref assign every agent $a \in M \cup W$ their respective list of agents of the opposite sex. This list pref(a) is ordered in strict order, ordered from the most preferred partner to the least preferred one, using the preference relation $\succ_a$.

An instance of the Stable Marriage can be interpreted as a bipartite graph, where every vertex has a strict ranking of it's incidental edges. Therefore each edge has two ranks, one from each neighboring vertice.

Example 1. Set of 2 men $M = \{m_1, m_2\}$, set of 3 women $W = \{w_1, w_2, w_3\}$ and preference relations $\succ = (\succ_{m_1}, \succ_{m_2}, \succ_{w_1}, \succ_{w_2}, \succ_{w_3})$ defined as:

$$w_2 \succ_{m_1} w_1 \succ_{m_1} w_3$$
$$w_1 \succ_{m_2} w_2 \succ_{m_2} w_3$$

$$m_1 \succ_{w_1} m_2$$
$$m_2 \succ_{w_2} m_1$$
$$m_2 \succ_{w_3} m_1$$

Tuple $I = (M, W, \succ)$ forms an instance of the Stable Marriage problem. Given relations $\succ$ make lists of preferences:

$$\text{pref}(m_1) = (w_2, w_1, w_3)$$
$$\text{pref}(m_2) = (w_1, w_2, w_3)$$

$$\text{pref}(w_1) = (m_1, m_2)$$
$$\text{pref}(w_2) = (m_2, m_1)$$
$$\text{pref}(w_3) = (m_2, m_1)$$

Definition 3 (Matching)**.** A *matching* Q in an instance I of Stable Marriage is a set of disjoint man-woman pairs $(m_i, w_j) \in Q$. Man m_i is said to be paired to woman w_j and woman w_j is said to be paired to man m_i. We write $Q(w_j) = m_i$ and $Q(m_i) = w_j$.

As such, matching in a Stable Marriage instance can be seen as a subset of edges in a bipartite graph, where each vertice appears at most once. Not every man and woman has to be included in a matching.

Example 2. If we take the instance from Example 1, then one of the possible matchings is $Q = \{(m_1, w_3), (m_2, w_1)\}$

Definition 4 (Preference)**.** If woman w_1 precedes woman w_2 in man m's preference list, then we say that man m prefers woman w_1 to woman w_2. We write $w_1 \succ_m w_2$.

Example 3. When we look at the man m_1 in an instance of Stable Marriage from Example 1, he prefers w_2 to w_1 and w_1 to w_3, so we can write

$$w_2 \succ_{m_1} w_1 \succ_{m_1} w_3$$

Now we are going to define what makes a marriage stable. For that we need to define about one more thing: *a blocking pair*.

Definition 5 (Blocking pair)**.** Given an instance of Stable Marriage I and a matching Q, a pair $(m_i, w_j), m_i \in M \wedge w_i \in W$, is *blocking pair* for Q if:

1. m_i is unassigned in Q or $w_j \succ_{m_i} Q(m_i)$, and
2. w_j is unassigned in Q or $m_i \succ_{w_j} Q(w_j)$.

Definition 6 (Stable matching)**.** Matching Q is said to be *stable* if it admits no blocking pair.

Example 4. Again, using the instance I from Example 1 and matching Q from Example 2, we can see that pairs (m_1, w_2) and (m_1, w_1) are both blocking pairs, as their members prefer each other over their current partners (in case of w_2 any partner is more preferred than no partner). Hence matching Q is not stable. If we look at a matching $Q' = (m_1, w_2), (m_2, w_1)$, we can check that it contains no blocking pairs. Therefore Q' is stable.

Theorem 1. Every instance I of Stable Marriage admits at least one stable matching.

This theorem was first shown in *College admissions and the stability of marriage* [1] by Gale and Shapley and they gave a constructive proof. Their algorithm, which is nowadays referred to as Gale-Shapley Algorithm or Deferred Acceptance Algorithm, finds a stable matching Q^* in polynomial time. We present this algorithm in the succeeding section.

1.3 Gale-Shapley algorithm

In this section, we introduce a polynomial-time algorithm which finds a *stable matching* for any Stable Marriage instance. We are also going to prove that the returned matching is truly stable. We define this algorithm formally in Algorithm 1, but for the general outline of the Gale-Shapley algorithm, we present it's two most important rules:

1. Every man *proposes* to the most preferred woman in his list of preferences. Every woman *temporarily pairs* with the man that is the highest ranked in her list of preferences and proposed to her. Every other man is *rejected*.

2. At the k-th step of the Gale-Shapley algorithm, those men that were rejected at $(k-1)$-th step propose to their *next most preferred woman*. Each woman temporarily pairs with the man that is the highest ranked in her list of preferences and proposed to her, or stays temporarily paired to the man she was paired to in the $(k-1)$-th step, if none of the new proposals is from a more preferred man. Every man that is not temporarily paired to some woman is *rejected*.

This algorithm is well-defined and terminates with the unique *man-optimal stable matching*. Termination can be easily proven, as each man proposes to each woman at most once. In fact, Gale-Shapley algorithm has at most $n^2 - 2n + 2$ stages, as proven by the upcoming Lemma 1, taken from [1], with our own proof, as it was without one.

Algorithm 1 Gale-Shapley algorithm

```
1:  function GaleShapleyAlg(I: instance of SM)
2:      M_f ← I.men()
3:      pref ← I.preferences()
4:      engagements ← []
5:      while M do
6:          m ← M_f.pop(0)
7:          w ← pref(m).pop(0)
8:          if w not in engagements then
9:              engagements[w] ← m
10:         else
11:             m2 ← engagements[w]
12:             if pref[w].indexof(m) < pref[w].indexof(m2) then
13:                 engagements[w] ← m
14:                 M_f.append(m2)
15:             else
16:                 M_f.append(m)
        return engagements
```

Lemma 1. Gale-Shapley algorithm has at most $n^2 - 2n + 2$ stages.

Proof. Every man can be rejected at most $n-1$ times, one of those times had to be in the initial stage though. In every stage that is not final, at least one man had to be rejected. Therefore one initial stage, one final stage and at most $n(n-2)$ times was some man rejected in other than the initial stage. This summed up gives us the promised $n^2 - 2n + 2$ stages. □

We claim that the output matching Q is stable. If not, there would be a blocking pair (m_i, w_j) such that:

1. $w_j \succ_{m_i} Q(m_i)$, and
2. $m_i \succ_{w_j} Q(w_j)$

That this cannot happen is proved in the upcoming lemma, based on Theorem 1 from [2]:

Lemma 2. Let Q be a matching created by the Gale-Shapley algorithm for an instance $I = (M, W, \succ)$ of the Stable Marriage problem. Then for all men $m \in M$ and all women $w \in W$:

$$Q(m) \succ_m w \vee Q(w) \succ_w m$$

Proof. Let (m_i, w_j) be a pair in the matching Q created by Gale-Shapley algorithm where $w_j \succ_{m_i} Q(m_i) \wedge m_i \succ_{w_j} Q(w_j)$. But man m_i must have

either proposed to w_j in an earlier step of algorithm and he was subsequently rejected by w_j in favor of some man that is more preferable for w_j, or m_i haven't proposed to w_i and therefore is paired with a more preferable woman than w_j, therefore they also cannot form a blocking pair where both of these conditions are fulfilled. □

Lemma 3. Let Q be a matching created by the Gale-Shapley algorithm for an instance I of Stable Marriage problem. Then we call Q the *man-optimal stable matching*. Meaning of man-optimal is that every man is paired with the most preferred woman, that he can be paired with, in all of the stable matchings in I. At the same time, all women are paired with their least preferable man that they can be paired with, in all of the stable matchings in the given instance of Stable Marriage I.

Proof. Let Q be a matching constructed by Gale-Shapley algorithm and $(m, w) \in Q$. During the computation, man m only proposes after being rejected by their current partner, except for the first proposal. Therefore man m was rejected by all more preferred women than w.

On the other hand, woman w can only get a better partner during the computation, since accepting her first proposal, as she is only rejecting her partner if a better proposal comes her way. Now we need to prove that none of the women that m was rejected by and none of the men that w rejected are matched with them in some stable matching.

Let's assume that there exists a woman w^* that is matched with m in a stable matching M^*. But then $Q(w^*) \succ_{w^*} m$, as m was necessarily rejected by w^* because she has got a better proposal from a more preferred partner. This would mean that $(Q(w^*), w^*)$ is a blocking pair for M^* and it is not stable.

In a similar fashion, let's suppose that woman w is matched with man m' whom she prefers less than m in some stable matching Q'. That would mean that w rejected man m and he is matched with some woman w'', $w \succ_m w''$. But this means that (m, w) is a blocking pair for Q'. □

The proof of the last lemma is ours. If we reverse the roles of men and women in proposing, then we get *woman-optimal stable matching*. Now, men are matched to their least preferred women and women are matched with their most preferred men that they can be paired with in any stable matching in a given instance of Stable Marriage.

In the case where the man-optimal and the woman-optimal stable matching is the same, then there exists only one stable matching for a given instance of Stable Marriage. But there is possibly even more stable matchings in one instance, when these two matchings are different. Later, in the Section 1.5, we research approximations of the count of stable matchings in one instance. We are going to explore how to enumerate them all in the next chapter.

1.4 Random Paths to Stability

In his paper, "Random Paths to Stability" [5], Alvin E. Roth discusses the application of random paths to solving the Stable Marriage problem (this section is based on the said paper). Roth presents a method for using a randomized algorithm to find a stable matching in a finite number of steps. The main principle of this approach is satisfying randomly chosen blocking pairs in a randomly chosen initial matching. We are going to describe the main theorem of said paper and the main technique used in its proof.

Theorem 2. Let Q be an arbitrary matching for a Stable Marriage instance $I = (M, W, \succ)$. Then there exists a finite sequence of matchings $Q_1, \ldots,$ Q_k, such that $Q = Q_1$, Q_k is stable, and for each $i \in \{1, \ldots, k-1\}$, there is a blocking pair (m_i, w_i) for Q_i such that Q_{i+1} is obtained from Q_i by resolving the blocking pair (m_i, w_i).

Proof. The proof works with k subsets $A_i \subseteq M \times W$, where $i \in \{1, \ldots, k\}$ and $A_1 \subset \cdots \subset A_k$. In every matching Q_i, there is a subset of agents A_i that does not contain blocking pairs. We begin this in Q_1 by randomly choosing a blocking pair (m_1, w_1) that creates the first set $A_1 = \{m_1, w_1\}$, as it alone cannot create any blocking pair in A_1.

Next, we continue inductively. A matching Q_q, $q \in \{1, k\}$ contains a subset of agents A_q without any blocking pairs.

If Q_q does not contain any blocking pairs, $k = q$ and we end here. Else there exists a blocking pair (m_{q+1}, w_{q+1}). At most one of these agents is contained in A_q, which splits this problem into three cases. Regardless on the case, we build the next subset of agents A_{q+1} as $A_{q+1} = A_q \cup \{m_{q+1}, w_{q+1}\}$.

If $|A_{q+1}| = |A_q|+1$, then the new agent (in this example we show woman w_{q+1} as the new agent, in the other case the roles of men and women are just reversed) chooses it's most preferred partner $m'_{q+1} \in A_{q+1}$ out of all the blocking pairs it is involved in. If the most preferred partner m'_{q+1} was unmatched in A_q, the new subset A_{q+1} is without blocking pairs.

Otherwise this new matching might have introduced a new blocking pair (m_{q+2}, w_{q+2}), where both $w_{q+2} = Q_q(m_{q+1})$ and m_{q+2} are contained in A_{q+1}. Again, we select the most preferred partner m'_{q+2} of w_{q+2} from all the partners in A_{q+1} that make a blocking pair with w_{q+2} and we create a pair (m'_{q+2}, w_{q+2}).

This process continues until we find ourselves with a matching such that no blocking pairs are contained in A_{q+1}, we label this matching as Q_{q+1}. This has to happen eventually, since no man ever matches with a less preferred woman and ,therefore, no blocking pair can appear twice in the sequence of their elimination. The set A_{q+1} is the set we require for the induction, as it strictly contains A_q and contains no blocking pair for Q_{q+1}.

In the case $|A_{q+1}| = |A_q| + 2$ where we add both agents from the selected blocking pair (m_{q+1}, w_{q+1}), we pair these two agents and the new subset A_{q+1}

is without blocking pairs. Again, set A_{q+1} conforms to our requirements, so it is the required set in this case.

This process must end eventually. The set A_q increases it's size strictly until a stable matching is reached. Also it cannot contain more agents than $M \cup W$. With this, the proof is now complete. □

Corollary 1. For any initial matching Q, the random process beginning by selecting an arbitrary matching Q and then proceeding to generate a sequence of matchings $Q_1, \ldots$, where each Q_{i+1} is generated by satisfying a single randomly chosen blocking pair in Q_i converges with probability of one to a stable matching. We assume a positive probability of choosing any particular blocking pair (m, w) in Q_i to be used to generate Q_{i+1} and depends only on the matching Q_i.

An interesting property of this algorithm is that every stable matching can be reached by some sequence of matchings, if all agents in the initial matching are unmatched. Based on the order in which we satisfy blocking pairs, we can obtain different stable matchings.

For example, we can obtain the man-optimal stable matching by choosing the first set A_1 to be the set of all women. Then the sequence of matchings which occurs in the Gale-Shapley algorithm is precisely the sequence constructed in the Theorem 2. Matchings converge to the man-optimal stable matching.

1.5 Count of Stable Matchings

We know how to find a stable matching and that it always exist in an instance of Stable Marriage problem. Now, we can talk about how many of them can we expect based on the size of our instance.

The smallest amount of stable matchings is obviously one, in an instance, that allows a stable matching, where every agent is matched with their most preferred partner. For the maximum amount of stable matchings in an instance, we only know upper and lower bounds.

One easy lower bound for the count of stable matchings is $O(2^{n/2})$ by combining disjoint instances of size 2. Knuth [28] has shown that instance made of four men and four women has at most ten stable matchings, found by exhaustive search (this instance is shown in Example 9). In a specific family of instances described by Robert W. Irving and Paul Leather [33], there is

$$g(n) = 3\{g(\frac{n}{2})\}^2 - 2\{g(\frac{n}{4})\}^4, \quad n \geq 4 \wedge n = 2^k, \quad k \in \mathbb{N}$$

stable matchings for instances with n men and n women. This has been since shown by Knuth (via personal communication, as shown in [34]) that this

recursive formulation equates

$$g(n) \geq \frac{2.28^n}{(1+\sqrt{3})},$$

therefore the lower complexity bound is $\Omega(2.28^n)$. This lower bound was generalized for every value of n by Edward G. Thurber [35] as a slightly weaker complexity bound of

$$\Omega(\frac{2.28^n}{(1+\sqrt{3})^{\log_2 n+1}}).$$

The simplest and obvious upper bound for the count of stable matchings is $O(n!)$. A better upper bound $O(n!/2^n)$ was found by Georgios K. Stathopoulos [36] (by exploiting *rotations*, which are explained later in Section 2.4). In 2017, Anna R. Karlin et al. [37] managed to prove an exponential upper bound $O(2^{17n})$. The best currently known upper complexity bound $O(3.55^n)$ was found in 2021 by Cory Palmer and Dömötör Pálvölgyi [38].

Chapter 2

Finding all Stable matchings

There are situations when just one stable matching is not enough for our needs. We may have to find all of them, or we search for a matching with some special property.

Here we are going to describe algorithms that can help us find all stable matchings in any Stable Marriage instance. With them, we can find all *stable pairs* (pairs that are included in at least one stable matching), all *rotations*, and all *stable matchings*. But first, we are going to introduce definitions of some useful concepts. This chapter is based on literature by McVitie and Wilson [39], Irving and Leather [33], Knuth [28] and Gusfield [3].

2.1 Ordering stable matchings

First, we are going to explain a relation of *dominance* between stable matchings. This relation is going to help us with ordering stable matchings later.

Definition 7 (Dominance [3])**.** Let Q and Q' be two stable matchings in an instance I of Stable Marriage. Let $max_i(Q, Q')$ be the most preferred woman of a man m_i between his two assigned partners in Q and Q'. Then $max(Q, Q')$ is a mapping of each man m_i to $max_i(Q, Q')$. We say that matching Q *dominates* matching Q' (from the perspective of men), if and only if $Q = max(Q, Q')$. Matching Q'' is *between* Q and Q' if and only if Q dominates Q'' and Q'' dominates Q'.

A matching dominating another matching is essentially a matching, where all men have the same or more preferred partner. Hence, there is not a men who has got a worse partner.

We define dominance from the viewpoint of men, as all of the literature we use also defines it this way. Obviously, it can also be defined from the viewpoint of women, which just reverses the order of dominance.

Example 5. Let's continue with the Stable Marriage problem instance I from Example 1. If we take two stable matchings $M_1 = \{(m_1, w_2), (m_2, w_1)\}$ and $M_2 = \{(m_1, w_1), (m_2, w_2)\}$, then the matching M_1 dominates the matching M_2.

Observation 1 (About dominance [3])**.** Let Q and Q' be two distinct stable matchings in an instance I of Stable Marriage. Let m be a man in I. Then:

- Both $max(Q, Q')$ and $min(Q, Q')$ are stable matchings.
- The unique maximum (most dominant) is *man-optimal stable matching*, and the unique minimum (least dominant) is *woman-optimal stable matching*.
- There is no stable matching, in which m is matched to a woman he prefers to his partner in the man-optimal marriage.

2.2 Break-marriage

Useful concept that will help us get a stable matching other than man/woman-optimal stable matching is breaking up a selected pair and resuming Gale-Shapley algorithm. We explore this process in this section.

Definition 8 (Breakmarriage [3, 39])**.** Let Q be a stable matching. Let man m be matched (paired) in Q to woman w. The operation `breakmarriage`(Q, m), is defined as:

1. Men and women are paired as in Q.
2. Selected pair (m, w) is broken up, making both m and w free
3. Restart Gale-Shapley algorithm, with man m now proposing to the next most preferred woman on his preference list he has not yet proposed to.
4. Operation terminates when either:
    - Some man has been rejected by all women
    - Everyone is paired (this happens after w is proposed to)

During the entire run of `breakmarriage`(Q, m) there is exactly one free man at any time. Hence, the sequence of proposals is completely determined: the next proposal is always made by the unique free man. It is also easy to see that Q dominates every stable matching that is constructed by applying *breakmarriage* operation on it.

Lemma 4 ([39])**.** If `breakmarriage`(Q, m) terminates with all men matched, then matched pairs form a stable matching.

Proof. All pairs are stable, if they were unaffected by the *breakmarriage* operation, since they were stable in the previous stable matching. If they were affected by the *breakmarriage* operation, and man m is matched to woman w at the end, while he prefers woman w' to w, then he must have had proposed to her but she rejected him. Because during the operation, all women get equal or better partner, hence woman w' is paired with a more preferred man than m, therefore the matching is stable. □

Theorem 3 ([39])**.** Every stable matching Q' can be obtained by a series of breakmarriage operations starting from the man-optimal matching Q_0.

Proof. Man-optimal matching Q_0 dominates every other stable matching Q' created by using `breakmarriage`(Q, m) operation, as no man can get a more preferred partner. Man-optimal stable matching is also the unique maximum – most dominant stable matching.

Woman-optimal stable matching is dominated by every other matching, as it is the unique minimum – least dominant stable matching. Therefore man-optimal stable matching Q_0 dominates woman-optimal stable matching and every matching between them.

When man m has a different partner in a stable matching Q than in Q' and Q dominates Q', then operation `breakmarriage`(Q, m) either results in Q' or in a stable matching between them (i.e. `breakmarriage`(Q, m) does not move any man to a woman below his proper partner in Q'). Hence any Q' can be derived from Q_0 by successively (and arbitrarily) choosing a man who isn't yet paired to his partner in Q'. □

This is a key theorem for finding stable matchings, for which we created our own simpler proof using domination relation, rather than the original proof by McVitie and Wilson [39]. Because the path towards getting stable matching Q' from man-optimal stable matching Q_0 is not completely deterministic as we are free in our choice of pair that we want to break by `breakmarriage`(Q, m) at each step, we are going to be always choosing the first man who is not yet paired to his intended partner in Q'.

Theorem 3 has two notable corollaries:

Corollary 2 ([3])**.** If `breakmarriage`(Q, m) results in matching Q', then Q' dominates all matchings which are dominated by Q and in which m is not paired to his partner in Q.

Corollary 3 ([3])**.** If m is paired to a woman other than his partner in the woman-optimal stable matching, then `breakmarriage`(Q, m) terminates with a new stable matching, i.e. no man is rejected by all the women.

Example 6. Let's consider a Stable Marriage instance with 3 men and 3 women with preferences as follows:

$$w_1 \succ_{m_1} w_2 \succ_{m_1} w_3$$
$$w_2 \succ_{m_2} w_3 \succ_{m_2} w_1$$
$$w_3 \succ_{m_3} w_1 \succ_{m_3} w_2$$

$$m_2 \succ_{w_1} m_3 \succ_{w_1} m_1$$
$$m_3 \succ_{w_2} m_1 \succ_{w_2} m_2$$
$$m_1 \succ_{w_3} m_2 \succ_{w_3} m_3$$

Man-optimal Stable Marriage for this instance is $Q_0 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. If we use `breakmarriage`$(Q_0, m_1)$, man m_1 breaks it's pair with woman w_1 and proposes to the next woman in his preference list, w_2.

Because $m_1 \succ_{w_2} Q_0(w_2) = m_2$, she accepts the proposal which frees the man m_2. Now m_2 proposes to w_3 according to his preference list and w_3 accepts the proposal according to hers.

This again frees man m_3, who's next most preferred partner is woman w_1, who is already free, ending this sequence of proposals with a new stable matching $Q_1 = \{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$. We can see that all men have a less preferred partner in matching Q_1 than in Q_0, therefore Q_0 dominates Q_1.

We can use *breakmarriage* once more on any pair in the new matching Q_1. This way we get the woman-optimal stable matching $Q_2 = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$. Now we can see that any additional *breakmarriage* operation is going to leave the chosen man without a partner, as his preference list would have been exhausted.

2.3 Finding all stable pairs in time $O(n^2)$

In this section, we are going to introduce the definition of a *stable pair*. We also describe a method to find all stable pairs for any instance of Stable Marriage problem.

Definition 9 (Stable pair [3])**.** Given an instance I of the Stable Marriage problem, a man-woman pair (m, w) is said to be a *stable pair* if and only if m is matched to w in some stable matching of I.

Example 7. Using the instance of Stable Marriage problem from Example 6, we can easily see that all possible pairs are actually stable pairs (as they all appear in the stable matchings Q_0, Q_1, Q_2 shown in the said example). That does not necessarily mean that using only stable pairs cannot create blocking pairs. For example, even though a matching $Q'' = \{(m_1, w_2),$

(m_2, w_1), $(m_3, w_3)\}$ is created purely from stable pairs, it is not stable, as a blocking pair (m_2, w_3) exists for it:

$$m_2 \succ_{w_3} Q''(w_3) = m_3 \wedge w_3 \succ_{m_2} Q''(m_2) = w_1.$$

Lemma 5 ([28])**.** Let I be an instance of Stable Marriage problem. The pair (m, w) is stable if and only if there is a stable matching Q in I omitting m and w, where each man who w prefers to m is matched to a woman he prefers to w, and where each woman who m prefers to w is matched to a man she prefers to m.

Proof. If that is not the case, then we are introducing a blocking pair by adding pair (m, w), which would mean that Q is not stable matching. □

The upcoming theorem is one of the most important pieces of knowledge from *Three Fast Algorithms for Four Problems in Stable Marriage* by Dan Gusfield [3]. It proves that we can find *all stable pairs* in any given instance using stable matchings and a dominance relation between them.

Theorem 4 ([3])**.** Let Q_0 and Q_t be the man-optimal and the woman-optimal matchings in an instance of Stable Marriage respectively. Let $Q_0, Q_1, \ldots, Q_t$ be a sequence of stable matchings such that for each $i \in [0, t-1]$, Q_i dominates Q_{i+1} and there is no stable matching between Q_i and Q_{i+1}. Then every pair appears in at least one of the marriages in the sequence.

Proof. Let Q_i and Q_{i+1} be two consecutive stable matchings of the sequence. Let m be matched to w_i in Q_i and to w_{i+1} in Q_{i+1}, $w_i \neq w_{i+1}$. From the dominance relation between matchings we know that $w_i \succ_m w_{i+1}$. If there exists a stable matching Q where m is paired to w, $w \neq w_{i+1} \wedge w \neq w_i$ and $w_i \succ_m w \succ_m w_{i+1}$. Then $Q' = min(Q_i, max(Q, Q_{i+1}))$ is also a stable matching. In Q', m is paired with w, hence it is different from both Q_i and Q_{i+1}. But then, since Q_i dominates Q' and Q' dominates Q_{i+1}, Q' is between Q_i and Q_{i+1}, a contradiction. □

Corollary 4 ([3])**.** Let H be the Hasse diagram of the lattice of all stable matchings in a Stable Marriage problem instance. Then the matchings along any path (directed by the dominance relation) in H between the man-optimal and woman-optimal marriages contain all the stable pairs.

The sequence of stable matchings $Q_0, Q_1, \ldots, Q_t$ for Theorem 4 have to be known to find all stable pairs. How to find those stable matchings is going to be shown later in Section 2.5, but first we introduce a concept of *rotations*.

2.4 Rotations

One of the most important definitions in the stable marriage theory is a *rotation*. Rotation is a tool to describe a difference between stable matchings in an instance of Stable Marriage. They show a "circle" in which partners move by one person to another (next preferred) to create a new stable matching. We have already seen rotations as an *all-or-nothing* cycle in Section 1.1.2, where it is used by Irving's algorithm [29] to find a stable matching in the stable roommates problem.

Definition 10 (Next preferred [3])**.** Let Q be a stable matching in an instance I of Stable Marriage. For any man m, let $S(m)$ be the first woman w' on m's list such that m prefers his partner $Q(m)$ to w' and w' prefers m to her partner $Q(w')$. Let $S'(m)$ be the man who $S(m)$ is matched to in Q.

Definition 11 (Rotation [33])**.** Let Q be a stable matching in an instance I of a Stable Marriage problem. Let $\pi = \{(m_1, w_1), \ldots, (m_s, w_s)\}$ be an ordered list of pairs from Q such that for each i from 1 to s, $S'(m_i) = m_{i+1 \bmod s}$. Then π is called a *rotation* (exposed in Q).

For any given matching there may be one, or many or even no (for woman-optimal stable matching) exposed rotations. Rotation is essentially describing which pairs are going to "rotate" their partners if we use `breakmarriage`(Q, m) operation on any one of the mentioned pairs. Next, we are going to define operations and an important theorem for rotations to make them useful in search for new stable matchings later.

Definition 12 (Moving [3])**.** For a given instance I of Stable Marriage problem, let π be a rotation exposed in stable matching Q, and let $Q(\pi)$ be the matching obtained by mating each man m in π with $S(m)$, and mating all men not in π with their partners in Q. We say that π *moves* each man and woman in π from their partners in Q to their partners in $Q(\pi)$. Rotation always moves a man to a less preferred partner, while always moving a woman to a more preferred partner. Moves by a rotation are independent of the marriage it is exposed in.

Definition 13 (Elimination [3, 33])**.** A pair (m, w), not necessarily stable, is said to be *eliminated* by rotation π if π moves w from m or below in her preference list to strictly above m. We say that we obtained $Q(\pi)$ by eliminating rotation π in Q.

Example 8. Again, using the Stable Marriage problem instance from Example 6, stable matching $Q_0 = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$ has one rotation $\pi = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$. If we eliminate this rotation, we get stable matching $Q_1 = \{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$, as we moved each man m to his $S(m)$ according to the rotation.

Theorem 5 ([33])**.** Except for the stable pairs that are part of women optimal matching (which are in no rotation), each stable pair is in exactly one rotation, and each pair in rotation is stable.

Proof. We refer to Irving [33]. □

This is another major theorem. The most important takeaway from it, except the fact that all stable pairs are contained in some rotation with the said exceptions, is that no stable pair is in two or more distinct rotations, hence there is only one deterministic option to eliminate a given stable pair (except for the order in which we eliminate rotations).

2.5 Pausing Breakmarriage Algorithm

Pausing Breakmarriage Algorithm (Algorithm 2), first presented by Gusfield in [3], is useful for creating a sequence of stable matchings from man-optimal to woman-optimal stable matching for any instance I of Stable Marriage problem. It computes man-optimal and woman-optimal stable matchings for I using Gale-Shapley algorithm. Then, it starts returning every stable matching in some sequence from man-optimal to woman-optimal (and in this order).

The key modification of the *breakmarriage* operation at the core of Algorithm 2 is the ability to pause the computation where the next stable matching in the sequence is output. There is a possibility that, during the run of a *breakmarriage* operation, some nested rotations can be eliminated.

Therefore, we might skip a stable matching that is between the original and the resulting stable matchings. The pause of Algorithm 2 occurs when the *nested rotation* π is recognized. Then the rotation π is returned and the next stable matching is generated from the previous matching by eliminating π.

By Corollary 3, each operation of *breakmarriage* is going to return a new stable matching (no man is going to be rejected by all women, as we are not breaking pairs that are found in the woman-optimal stable matching), although we still have to show stability of all matchings in the sequence and that we have found all stable matchings in the sequence. Before show the proof of correctness of the Algorithm 2, we are going to point out some observations about the actions of algorithm. These are going to simplify later proofs. Every observation, lemma and theorem (and their respective proofs, with the exception of proof of Theorem 7, which is ours) in this section is from Gusfield [3].

Observation 2. The men and women in $\{m, Q_k(m) : Q_k(m) \neq Q_{k+1}(m), m \in M\}$ are exactly the same men and women in the pairs of π_k. The men in π_k prefer their partners in Q_k to their partners in Q_{k+1} and women prefer their partners in Q_{k+1} to their partners in Q_k.

Algorithm 2 Pausing Breakmarriage Algorithm

```
1: function PausingBreakmarriage(I: instance of SM)
2:     i ← 0
3:     M ← I.men()
4:     W ← I.women()
5:     Q_0 ← GaleShapleyAlg(M, W)
6:     Q_t ← GaleShapleyAlg(W, M).swapPartities()
7:     marks ← []
8:     result ← {}
9:     while Q_i ≠ Q_t do
10:        marks ← {}
11:        m ← first({m' : m' ∈ M ∧ Q_i(m') ≠ Q_t(m')})
12:        Q ← Q_i
13:        w ← Q(m)
14:        marks ← marks ∪ [w]
15:        output, marks ← breakmarriagePausing(Q, m, marks, i)
16:        i ← i + 1
17:        result ← result ∪ output
18:        if Q_{i−1}(m) ≠ output[0][0][1] then
19:            Go to: 12
       return result ∪ Q_t

1: function breakmarriagePausing(Q, m, marks, i)
2:     m' ← m
3:     while S(m') not in marks do
4:         m' ← S'(m')
5:         marks ← marks ∪ [Q(m')]
6:     w' ← S(m')
7:     π(W) ← marks[indexof(w') :]
8:     π(M) ← [Q(w) for w in π(W)]
9:     π_i ← [(Q(w), w) for w in π(W)]
10:    Q_{i+1} ← (Q \ π_i) ∪ {(m'', S(m'')) for m'' in π_i}
11:    output ← π_i
12:    marks ← marks \ π(W)
13:    if Q(m) ≠ w' ∧ Q(w') ≻_{w'} m' then
14:        marks ← marks ∩ w'
       return output, marks
```

Observation 3. Let m_1 and m_2 be two different men in any two consecutive pairs of π_k. If they are matched to w_1 and w_2, respectively, in Q_k, then m_1 is matched to w_2 in Q_{k+1}.

Observation 4. Let m_1 and m_2 be two different men in any two consecutive pairs of π_k. Let them be matched to w_1 and w_2, respectively, in Q_k. Then w_2 is the first woman below w_1 on m_1's list such that w_2 prefers m_1 to m_2.

Lemma 6. Each Q_i found by Algorithm 2 is a stable matching.

Proof. It is proven by induction. Q_0 is stable, and we assume that all marriages up through Q_k are stable.

Suppose Q_{k+1} is not stable, then there is a man m and a woman w who block Q_{k+1}. Since, by Observation 2, each woman either improves in Q_{k+1} (over Q_k) or keeps her same partner, m must be in a pair in π_k, otherwise m and w would block Q_k.

For the same reason, m cannot prefer w to his partner in Q_k so w must be strictly between (in order of preference) m's partner in Q_k and his partner in Q_{k+1}; Let B denote these women. But by Observation 4, none of the women in B prefer m to their partners in Q_{k+1}. Hence Q_{k+1} is stable matching. □

Lemma 7. There is no stable matching between Q_k and Q_{k+1}.

Proof. Suppose, to the contrary, that Q is a stable matching between Q_k and Q_{k+1}. We claim that no man m can be paired in Q to a woman between his partner in Q_k and his partner in Q_{k+1}.

Let w be such a woman between m's respective partners, and let m_w be the partner of w in Q_k. By Observation 4, w prefers m_w to m, and since Q_k dominates Q and m_w is not paired with w in Q, m_w prefers w to his partner in Q. Hence m_w and w block Q.

So if a stable matching Q exists between Q_k and Q_{k+1}, then every man is either paired to his partner in Q_k or to his partner in Q_{k+1}, and there must be at least one man of each type (else Q is equal to either Q_k or Q_{k+1}). Now in the (circular) order of pairs given in π_k, let m and m' be any two consecutive men in π_k, and let w and w' be their respective partners in Q_k. Recall (by Observation 3) that w' becomes the partner of m in Q_{k+1}.

Hence it is not possible that in Q, m pairs with his partner in Q_{k+1} and m' is paired with his partner in Q_k, since they both would then marry w'. Similarly, it is not possible that in Q, m pairs with his partner in Q_k, since they both would then match w'. Similarly, it is not possible that in Q, m pairs with his partner in Q_k and m' pairs with his partner in Q_{k+1}, for then w' would be unpaired in Q.

But then either $Q = Q_k$ or $Q = Q_{k+1}$. Hence there is no stable matching Q between Q_k and Q_{k+1}. □

Theorem 6. The matchings found by the Algorithm 2 contain all stable pairs.

Proof. This theorem is a corollary of Lemmas 6 and 7 and Theorem 4. Let I be an instance of Stable Marriage problem. Thanks to both lemmas we know that we have found a sequence of stable matchings from man-optimal to woman-optimal stable matching. From the theorem, we know that every stable pair of I is appears in at least one of these matchings. □

Theorem 7. All stable pairs can be found and output by the Algorithm 2 in $O(n^2)$ time.

Proof. Man-optimal and woman-optimal stable matchings are computed by Gale-Shapley algorithm, and therefore their complexity is $O(n^2)$. Since no man-woman pair is in more than one rotation and since no man proposes to a woman from his list more than once, the time complexity is also $O(n^2)$.

We output only all of the rotations plus Q_t to efficiently represent the found stable matchings. Output contains every stable pair and we output every stable pair only once, therefore even the output has a complexity of $O(n^2)$. □

2.6 Finding all rotations in $O(n^2)$

Each cycle found by Algorithm 2 is clearly a rotation. Hence the Algorithm 2 finds a set of distinct rotations that contain all stable pairs other than those contained in woman-optimal matching. Therefore, by Theorem 5, Algorithm 2 finds all rotations, and outputs each one exactly once (as shown in the last section), and so all rotations can be found and output in $O(n^2)$ time. [3]

Theorem 8 ([3])**.** Let P be any path in the Hasse diagram H from man-optimal to woman-optimal matching. Then any two consecutive matchings on P differ by a single rotation. Set of rotations between matchings along P contain all rotations exactly once.

Proof. Corollary 4 says that all stable pairs are contained in any path in H between the man-optimal and woman-optimal stable matching. Thanks to the Theorem 5 we know that every stable pair except for the ones contained in the woman-optimal stable matching are contained in exactly one rotation. And we know that Algorithm 2 returns every rotation. Hence now it's only a corollary of Theorems 6 and 7. □

It is now easy to see that a sequence of stable matchings from man-optimal to woman-optimal matchings, which satisfy the conditions of Theorem 4, must lie on such a path P in H. Therefore, Pausing Breakmarriage Algorithm enumerates the matchings along some path P in H from man-optimal to woman-optimal matching.

2.7 Enumerating all stable matchings

In this section, we describe algorithm enumerating all stable matchings in $O(n^2 + n|S|)$ time and $O(n^2)$ space, found by Gusfield [3], where S is the set of all stable matchings in a given instance of Stable Marriage problem. In the same time bound, we can explicitly construct the Hasse diagram of the lattice of all stable matchings.

First, we need to introduce definitions about *partial order* of rotations. From this order, we are going to define a directed graph of rotations (*rotation poset*), and later it's sparse subgraph. Later, we present a tree structure built using the sparse subgraph of rotations and a method to enumerate all stable matchings from it.

Definition 14 (Precedence [33])**.** Let π and ρ be two distinct rotations. Rotation π is said to *explicitly precede* ρ if and only if π eliminates a pair (m, w), and ρ moves m to a woman w' such that m prefers w to w'. The relation *precedes* is defined as the transitive closure of the relation "explicitly precedes".

Precedence relation means that no matter how the men are ordered, Pausing Breakmarriage Algorithm always finds rotation π before rotation ρ. It is easy to see that if a rotation $\pi = \{\dots, (m, w), (m', w'), (m'', w''), \dots\}$ exists, a rotation $\rho = \{\dots, (m, w'), (m', w''), \dots\}$ can only be exposed *after* the rotation π has been eliminated, as every pair appears in only one rotation. Precedence also defines a partial order among rotations. The following definitions are using precedence as a partial order to construct graph and closed subset.

Definition 15 (Graph of rotations – rotation poset [3])**.** Given an instance of the Stable Marriage problem, let D be a directed acyclic graph, where the vertices of D are in one-to-one correspondence with the set of rotations (we name each vertex after its corresponding rotation) and for any two vertices π and ρ, there is a directed edge from π to ρ if and only if rotation π precedes rotation ρ.

Note that D may have $\Theta(n^2)$ vertices and $\Theta(n^4)$ edges [3]. It is also important to note that in this definition by Gusfield [3], edges are between the rotations, where one *precedes* the other, while Irving and Leather [33] define the edges only between the vertices, where one *explicitly precedes* the other.

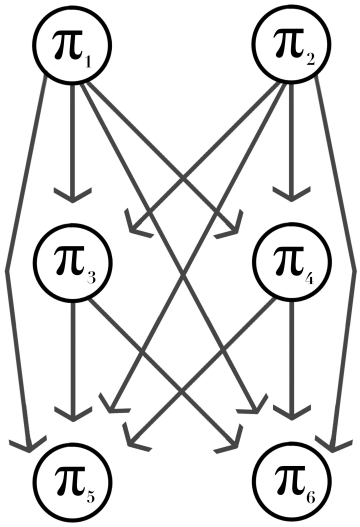

Figure 2.1: Graph D for example 9

Example 9. Let's take instance I of the Stable Marriage problem of size four, with preferences:

$$w_1 \succ_{m_1} w_2 \succ_{m_1} w_3 \succ_{m_1} w_4$$
$$w_2 \succ_{m_2} w_1 \succ_{m_2} w_4 \succ_{m_2} w_3$$
$$w_3 \succ_{m_3} w_4 \succ_{m_3} w_1 \succ_{m_3} w_2$$
$$w_4 \succ_{m_4} w_3 \succ_{m_4} w_2 \succ_{m_4} w_1$$

$$m_4 \succ_{w_1} m_3 \succ_{w_1} m_2 \succ_{w_1} m_1$$
$$m_3 \succ_{w_2} m_4 \succ_{w_2} m_1 \succ_{w_2} m_2$$
$$m_2 \succ_{w_3} m_1 \succ_{w_3} m_4 \succ_{w_3} m_3$$
$$m_1 \succ_{w_4} m_2 \succ_{w_4} m_3 \succ_{w_4} m_4$$

This instance is actually the instance with the most stable matchings for any instance of size four, as shown by Knuth [28]. We are going to show the stable matchings in "layers" based on the precedence of rotations that are required to reach them from the man-optimal stable matching. The first stable matchings are:

$$\{(m_1, w_1), (m_2, w_2), (m_3, w_3), (m_4, w_4)\}$$
$$\{(m_1, w_2), (m_2, w_1), (m_3, w_3), (m_4, w_4)\}$$
$$\{(m_1, w_1), (m_2, w_2), (m_3, w_4), (m_4, w_3)\}$$
$$\{(m_1, w_2), (m_2, w_1), (m_3, w_4), (m_4, w_3)\}$$

These are man-optimal stable matching, and stable matchings created from it by eliminating rotations $\pi_1 = \{(m_1, w_1), (m_2, w_2)\}$, or $\pi_2 = \{(m_3, w_3), (m_4, w_4)\}$. These rotations precede all other rotations.

$$\{(m_1, w_2), (m_2, w_4), (m_3, w_1), (m_4, w_3)\}$$
$$\{(m_1, w_3), (m_2, w_1), (m_3, w_4), (m_4, w_2)\}$$
$$\{(m_1, w_3), (m_2, w_4), (m_3, w_1), (m_4, w_2)\}$$

On top of eliminating both preceding rotations, these stable matchings also eliminate newly exposed rotations $\pi_3 = \{(m_2, w_1), (m_3, w_4)\}$, or $\pi_4 = \{(m_1, w_2), (m_4, w_3)\}$. Again these rotations precede all of the upcoming rotations.

$$\{(m_1, w_3), (m_2, w_4), (m_3, w_2), (m_4, w_1)\}$$
$$\{(m_1, w_4), (m_2, w_3), (m_3, w_1), (m_4, w_2)\}$$
$$\{(m_1, w_4), (m_2, w_3), (m_3, w_2), (m_4, w_1)\}$$

The last set of rotations $\pi_5 = \{(m_1, w_3), (m_2, w_4)\}$, and $\pi_6 = \{(m_3, w_1), (m_4, w_2)\}$ can be exposed after eliminating the last exposed set of rotations, and by eliminating the rotation π_5 or π_6, we get the last three stable matchings for our instance I.

The rotation poset D for I (graph shown on Image 2.1) has necessarily six vertices, one for each rotation. Every rotation π_i precedes every rotation π_j, where $j > i + (i \bmod 2)$, which means that D has twelve edges.

Definition 16 (Closed subset of rotations [3])**.** A subset of rotations SN of D is *closed* if and only if SN contains all rotations which precede the rotations in SN.

Irving and Leather [33] named this closed subset of rotations as *antichain*. Their definition is equivalent to this one from Gusfield [3] we used.

Theorem 9 ([33])**.** Let S be the set of all stable matchings for a given instance of the Stable Marriage problem. Let D be the corresponding directed graph formed from the set of all rotations. Then there exists a one-to-one correspondence between S and the family of closed subsets in D, i.e. each closed subset in D specifies a distinct stable matching, and all stable matchings are specified in this way.

Proof. We refer to [33] for proof. □

Any stable matching can be reconstructed by starting with the man-optimal stable matching and applying rotations from it's closed subset of

rotations SN. They must be applied in any order consistent with their precedence relation, meaning that the moves of rotation π can be made only after the moves of all rotations which precede π.

Enumerating each closed subset of D to get all stable matchings is not going to give us the promised $O(n^2 + n|S|)$ time and $O(n^2)$ space complexity, as D itself needs $\Theta(n^4)$ space to store, hence it could not be achieved using D. The main idea of the speedup is to use a sparse subgraph of D which preserves all the closed subsets, and which can be built quickly.

Any subgraph of D, whose transitive closure is D, preserves the closed subsets. We can construct such a subgraph G, in $O(n^2)$ time, with the property that G has $O(n^2)$ edges, and that no vertex in G has out degree more than n. This bounded out degree is one of the keys to the $n|S|$ term in the time bound, the sparsity is integral to the space bound. Now we are going to define rules for such sparse subgraph and then prove some of it's traits.

Definition 17 (Subgraph of rotations [3])**.** Let D be a directed graph of rotations of an instance of stable matching problem. Let G be a directed acyclic subgraph of D containing all vertices of D but only the edges which comply with one of these two rules, which are applied for each man m whose partner in man-optimal stable matching is is different than his partner in woman-optimal stable matching:

1. Let $W(m) = \{w_0, \ldots, w_r\}$ be the set of women in decreasing order of preference by m, such that $\forall i \in \widehat{r} = \{0, \ldots, r\}$, (m, w_i) is a stable pair. For $\forall i \in \widehat{r-1}$, let π_i be the rotation containing pair (m, w_i), and let $\Pi(m)$ be the set of these rotations. then, for $\forall i \in \widehat{r-2}$, G contains an edge from π_i to π_{i+1}.

2. Let $W(m) = \{w_0, \ldots, w_r\}$ be the set of women in decreasing order of preference by m, such that $\forall i \in \widehat{r}$, (m, w_i) is a stable pair. Suppose (m, w) is a non-stable pair eliminated by a rotation π, such that m prefers w to any other woman w' in any other pair (m, w') eliminated by π. If there are women w_i and w_{i+1} in $W(m)$ such that m prefers w_i to w and m prefers w to w_{i+1}, then G contains an edge from π to π_i.

Lemma 8 ([3])**.** Let D be a directed graph of rotations of an instance of stable matching problem. Let G be a directed acyclic subgraph of D. Then G has only $O(n^2)$ edges. It can be constructed in $O(n^2)$ time and no vertex in G has out degree more than n.

Proof. The rotations can be found in $O(n^2)$ using Algorithm 2. We label each pair that is eliminated by some rotation with the name of the eliminating rotation. To do this, we examine each rotation π, and, for each woman w in a pair in π, we note the men that π moves w over, each of these pairs is labeled with π. Since π eliminates a set of pairs corresponding to a contiguous

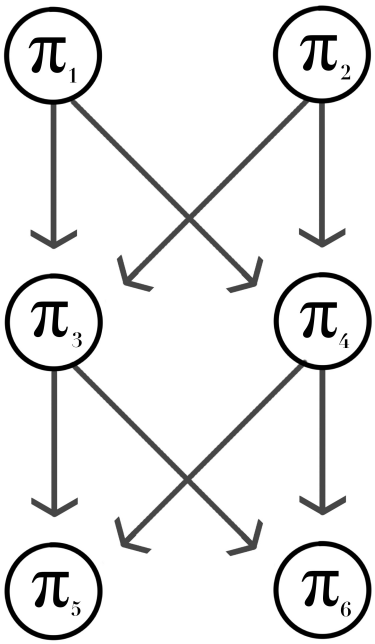

Figure 2.2: Graph G for example 10

sequence of men in w's preference list, finding these pairs takes constant time per pair. Since no pair is eliminated by more than one rotation, these labelings can be done in $O(n^2)$ time.

Now, G can be constructed by processing each man m's list top down, keeping a mark on the most recently encountered stable pair in m's list. When a new stable pair is encountered, we create an edge in G from the rotation labeling the marked pair (if there is one) to the rotation labeling the new pair, and we update the mark. When a non-stable pair is encountered, we check (in unit time using a random access list of the rotations) if its label has already been encountered in m's list. If not, then we create an edge in G from the rotation labeling the marked pair (there will be one) to the rotation labeling the current non-stable pair. Each scan down a man's list takes $O(n)$ time, hence $O(n^2)$ time in total.

Since the total time to build G is $O(n^2)$, it can only have $O(n^2)$ edges. From the details above, it is also clear that for any rotation π, the scan down a given man m's list adds at most one edge out of π, hence the out degree of any vertex in G is bounded by n, the number of men. □

Example 10. If we continue from our Example 9 and build the subgraph of D called G, first thing we can see is that every pair in I is stable, hence the second rule for creating an edge in G is not going to be applied. We are going to create sets $\Pi(m)$ for every man m, according to it's definition:

$$\Pi(m_1) = \{Q_1, Q_4, Q_5\}$$
$$\Pi(m_2) = \{Q_1, Q_3, Q_5\}$$
$$\Pi(m_3) = \{Q_2, Q_3, Q_6\}$$
$$\Pi(m_4) = \{Q_2, Q_4, Q_6\}$$

It's easy to create graph (shown on Image 2.2), when we have these sets. G has six vertices again, but the edges have been reduced from twelve to eight.

Lemma 9 ([3])**.** Let D be a directed graph of rotations of an instance of stable matching problem. Let G be a directed acyclic subgraph of D. For any two rotations π and ρ, π precedes ρ if and only if π reaches ρ by a directed path in G, hence the transitive closure of G is D, and so the closed sets of G and D are identical.

Proof. G is a subgraph of D since each edge in G specifies a precedence relation between the rotations at the endpoints of the edge. To prove the other direction, it suffices to show that if π explicitly precedes ρ then π reaches ρ in G. By definition of "explicitly precedes", there must be two women w and w^* such that (m, w) is eliminated by π, and ρ moves m to w^*, and m prefers w to w^*. Then w^* is in $W(m)$, and $\rho \in \Pi(m)$; say $\rho = \pi_{i^*}$, where, by definition, ρ moves m from w_{i^*} to w^*. So in G there is a directed path from π_i to ρ for every $\pi_i \in \Pi(m)$ such that $i \leq i^*$.

Now let w' be the woman most preferred by m such that (m, w') is eliminated by π. By construction of G, there is an edge (associated with the pair (m, w')) from π to $\pi_{i'}$, for some $\pi_{i'} \in \Pi(m)$; let $w_{i'}$ be the woman that $\pi_{i'}$ moves m from. So if $i' \leq i^*$ (i.e. $w_{i'}$ is equal to or is preferred to w_{i^*}), then there is a directed path in G from π to ρ. But m prefers $w_{i'}$ to w, and w^* to w, and since, by the actions of Algorithm 2, man m is moved over any particular woman by at most one rotation, w_{i^*} cannot be preferred to w'; hence $w_{i'}$ must either be w_{i^*} or be preferred to w_{i^*}, and the lemma follows. □

G is not necessarily the transitive reduction of D, as general algorithms to produce the transitive reduction would take much more time than the $O(n^2)$ time to construct G. However, G is sufficient for our time and space complexities.

2.7.1 The enumeration algorithm

This algorithm uses G to build a tree T, which we are going to describe. Tree T with root r has every edge labeled with a rotation such that the path from the root r to any vertex in T enumerates a distinct closed set SN of rotations in G, and such that each closed set in G is enumerated in this way.

For any rotation π on a given edge $e = \{x, y\}$, all rotations that precede π will be on the path from the root r to x. It follows inductively that the stable matching corresponding to any vertex x can be explicitly constructed by starting at the root and successively executing the moves dictated by each rotation on the path to x.

Each change takes $O(n)$ time, and each vertex in T corresponds to a distinct stable matching. It follows that all the stable matchings can be output in $O(n)$ time per marriage, once T has been constructed. If T is traversed depth first, then only one complete marriage must be known at any time, hence only $O(n)$ additional space is needed for the traversal of T. [3]

2.7.1.1 Build of tree T

This paragraph is going to talk about a way to build the tree T. The idea of it's construction is going to be shown first, and then, also in the role of a time complexity proof, we are going to explain the construction in more detail.

First, a numerical labeling of rotations is needed. We label them in order of the topological ordering of G, therefore every rotation has a larger label than any of it's predecessors. These labels can be found in linear time in the number of edges of G, that is, in $O(n^2)$ time.

For building the tree T, we start at the root r and successively expand from any unexpanded vertex in T as follows:

- Let $R(y)$ be the rotations along the path from r to y in T, and let $e = \{x, y\}$ be the last edge on this path.
- Let $MR(y)$ be the set of maximal rotations (vertices in G with indegree zero) when all the rotations in $\pi(y)$ are removed from G.
- Let $LR(y)$ be those rotations in $MR(y)$ whose label is larger than the label on edge e.

Then y is expanded by adding $|LR(y)|$ edges out of vertex y. Each one of those edges is labeled with a distinct rotation in $LR(y)$. [3]

Lemma 10 ([3])**.** Given G, T can be constructed in $O(n)$ time per vertex.

Proof. Let $e = \{x, y\}$ be the last edge on the path to y. Let the rotation on e be π. Let there be a graph $G(x)$ at vertex x in T, obtained from G by

deleting all vertices in $R(x)$ and all incident edges. Let all indegrees of all vertices in $G(x)$ be known.

Then $LR(y)$ is the set of all neighbors of π in $G(x)$ which have indegree one and (due to topological labeling) they all have a larger label than π, together with the set of rotations $LR(x)$, whose label is also larger than π. $LR(y)$ can be found in $O(n)$ time since no vertex in G (hence in $G(x)$) has out degree more than n (there are at most n neighbors of π in $G(x)$). For $LR(x)$ we also claim that $|LR(x)| \leq n$, because for any man m, if (m, w) and (m, w') are two distinct rotations, then clearly one of them must precede the other. By construction of induction, every pair of rotations from $LR(x)$ must be incomparable, and so any man m is in at most one pair in at most one rotation in $LR(x)$.

Constructing graphs at each of the endpoints must be done in depth first manner. If we constructed T in breath first manner, then $|LR(x)|$ graphs would have to be constructed and stored. If we expand a given vertex x in T depth first, we find all maximal elements in $G(x)$ and store them, essentially constructing all edges out of x, and we construct only one new graph $G(y)$ for only one chosen edge $\{x, y\}$ out of x. Graph $G(x)$ can be transformed into $G(y)$ in $O(n)$ time by deleting y and all incident edges from $G(x)$.

The indegree of each neighbor of x in $G(y)$ is one less than it's indegree in $G(x)$. All other indegrees remain as in $G(x)$. Hence the indegrees are also maintained in $O(n)$ time. The algorithm can now continue with expanding from y.

While backing up from y to x, we use $G(y)$ and the rotation on the edge $\{x, y\}$ to reconstruct $G(x)$ in time $O(n)$. Now with $G(x)$ and the unexpanded children vertices of x, we choose unexpanded child y' of x, and again we transform $G(x)$ into $G(y')$, and then expand y'. Therefore T can be built in $O(n)$ time per vertex. □

Lemma 10 shows that we can construct tree T in $O(n)$ per vertex, but does not show us how to enumerate all stable matchings. That is going to be shown in the proof of the upcoming corollary.

Corollary 5 ([3])**.** Given G, the set of all stable matchings can be enumerated in $O(n)$ time per marriage, and $O(n^2)$ total space.

Proof. The construction of tree T takes $O(n)$ time per vertex, but we don't need to build it explicitly. What is sufficient at any time is the path from r to the current vertex being expanded and the edges which directly hang off that path.

The depth of T is at most $O(n^2)$ and the outdegree of each vertex in T is $O(n)$, hence if we construct and output the stable matchings as T is being implicitly built depth first, then we need $O(n^3)$ space for T. However this space bound could have been reduced, if we hadn't stored the maximal elements of $G(x)$ at each vertex x. They can be found when backing up from vertex y

to x, where $\{x, y\}$ is labeled with rotation π, the maximal elements of $G(x)$ can be reconstructed from $G(y)$ in $O(n)$ time, since they are the maximal elements of $G(y)$, plus π, minus the neighbors of π in $G(x)$.

So both $G(x)$ and its maximal elements can be recomputed in $O(n)$ time on backup. However, we must care for not traversing any edge out of x more than once.

A simple way to do this is to traverse the edges out of x in increasing order of their labels. Each time we enter a vertex x to expand, we scan the maximal elements of $G(x)$ and choose the one with the smallest label larger than the label on the edge just used to enter x (either backup or first entry).

In this way only a single path of tree T needs to be kept at any one time, hence the total space is $O(n^2)$, and the time remains $O(n)$ per vertex. □

The only thing that is needed to be shown now is that the vertices in T are in one-to-one correspondence to the closed sets of G, and that the order of the rotations along a path in G has the desired properties claimed above. This is going to be done in the following lemmas.

Lemma 11 ([3])**.** Let x be an arbitrary vertex in T. Then $R(x)$ is a closed set of rotations in D (hence G).

Proof. By induction, this lemma is clearly true for the root r of T, which corresponds to the empty set, and for vertices at distance one from the root, for each of these correspond to a maximal rotation in D. Let x be a vertex. Let $\{x, y\}$ be an edge out of x with label π.

From inductive hypothesis, $R(x)$ is a closed set, and, by construction, π is maximal in $G(x)$, so all the predecessors of π are in $R(x)$. Hence $R(x) + \{\pi\}$ is closed set in D, and this set corresponds to vertex y. □

Lemma 12 ([3])**.** Every closed set in D is $R(x)$ for some vertex x in T.

Proof. Again we prove this lemma by induction, now on the size of the set. For size zero and one, the lemma is clearly true, as these sets are the empty set and the maximal elements in D.

Now let SN be a closed set of size $k+1$. SN must have a minimal element with respect to the partial order D. Let π be the minimal element of SN with the largest label.

By the induction hypothesis, $SN - \{\pi\}$ is $R(x)$ for some vertex x in T. But, then π is a maximal vertex in $G(x)$, and since it has the largest label of the rotations in SN, it will label an edge $\{x, y\}$ out of x. Hence, SN is $R(y)$. □

Lemma 13 ([3])**.** Let x and x' be two distinct vertices in T, then $R(x) \neq R(x')$, hence no closed set is enumerated twice in T.

Proof. Consider a vertex x and two edges $\{x, y\}$ and $\{x, y'\}$ out of x labeled π, resp. ρ, where π has a smaller label than ρ. Note that all the labels along any

path from r are in increasing order, hence π cannot appear in the subtree of T rooted at y'. The lemma follows by applying this observation inductively on the length of the paths. □

With these lemmas, we can finally say that we can enumerate all stable matchings while implicitly building tree T, and that we find every stable matching exactly once. Now we finally have an algorithm for any instance of Stable Marriage that can output all stable matchings in $O(n^2 + n|S|)$ time and $O(n^2)$ space. From Section 1.5 we know that the best currently known upper bound for the number of stable matchings is $O(3.55^n)$, so we can say that the time complexity is $O(n^2 + n3.55^n)$.

Chapter 3

Algorithms using enumeration of stable matchings

We present a detailed theory of enumerating stable matchings, because many problems that are similar to the one we are facing also use this theory. If we were to find a solution of our problem using the tree of stable matchings as shown earlier in Section 2.7.1, we would be possibly considering a far smaller amount of matchings as our solution ($O(3.55^n)$ instead of $n!$ matchings).

These problems often try to build the tree of stable matchings implicitly depth first. At each vertex (stable matching) they deterministically choose an edge (rotation) by which to continue the algorithm and they usually terminate at the stable matching they seek, rather than scanning the whole path to the woman-optimal stable matching and choosing the stable matching best fitting the given condition after.

We are going to take a look at three problems using the lattice of stable matchings to find their solution. Not all of them construct the tree of stable matchings the same way as Gusfield [3] did, but all of them use similar techniques of exploiting the structure based on the rotation poset by Irving and Leather [33].

3.1 The minimum regret stable marriage

This problem is first presented by Knuth [28], among many other problems related to stable matchings. It tries to quantify a "*regret*" of each man and woman.

Regret in this context refers to the difference between the preference ranking of an agent's assigned partner and their most preferred partner, that they did not get matched with. The maximum regret is the highest value among all participants in the matching.

The Minimum Regret Stable Marriage problem seeks to find a match-

ing that minimizes this maximum regret value. Upcoming definitions are going to formally define *regret*, and the minimum regret stable matching. This section is based on articles [3, 28] by Gusfield and Knuth respectively.

Definition 18 (Regret of a person [28])**.** Let Q be a matching in an instance of Stable Marriage problem. Let (m, w) be a pair in the matching Q. The *regret* of m, denoted $\mathrm{r}(m)$, is the position of woman w in m's preference list $\mathrm{pref}(m)$, and the *regret* of w, denoted $\mathrm{r}(w)$, is the position of man m in w's preference list $\mathrm{pref}(w)$.

Definition 19 (Regret of a matching [28])**.** Let Q be a matching in an instance of Stable Marriage problem. The *regret* of a marriage Q, denoted $R(Q)$, is defined to be the maximum regret of any person, given the pairing in Q. Hence, Q is measured by the person who is worst off in it.

According to Gusfield [3], the solution given by Knuth [28] has a time complexity $O(n^4)$. Gusfield has shown that using the *breakmarriage* operation (Definition 8), we can obtain a method that runs in $O(n^2)$ time, which is allowed by avoiding duplicated proposals and rejections (specifically by using Corollary 2).

The problem can be split in two:

- *woman-regret minimum* – "find, if one exists, a marriage minimizing $R(Q)$ over all stable matchings in which at least one woman is a person with the maximum regret in the marriage"[3]
- *man-regret minimum* – "find, if one exists, a marriage minimizing $R(Q)$ over all stable matchings in which at least one man is a person with the maximum regret in the marriage"[3]

Both problems are effectively searching for the same value $R(Q)$, but they aren't necessarily returning the same matching, as more matchings with the same regret $R(Q)$ may exist. It is possible that for a given instance of Stable Marriage problem, there are no men (women) with maximum regret, and therefore no man-regret (woman-regret) minimum exists. This happens only if all the people with maximum regret in man (woman) optimal marriage are men (women). From now on, we assume that both woman-regret and man-regret minimum exists, since the case where they don't can be easily checked for as explained above. The Algorithm 3 below finds a woman-regret minimum, assuming both woman-regret and man-regret minimum exists. It can be easily modified to find a man-regret minimum.

Algorithm 3 Finding a woman regret minimum

```
1:  function Algorithm B(I: instance of SM)
2:      i ← 0
3:      M ← I.men()
4:      W ← I.women()
5:      Q_0 ← GaleShapleyAlg(M, W)
6:      Q_t ← GaleShapleyAlg(W, M).swapPartitions()
7:      i ← 0
8:      R_i ← W.where(lambda w : r(w, Q_i) == r(Q_i))
9:      while not empty(R_i) do
10:         w ← getFirst(R_i)
11:         m ← Q_i(w)
12:         if (m, w) in Q_t then return Q_i
13:         Q_{i+1} ← breakmarriage(Q_i, m)
14:         i ← i + 1
15:         smregrets_i ← W.where(lambda w : r(w, Q_i) == r(Q_i))
        return Q_{i-1}
```

The Algorithm 3 modifies Q_0 towards Q_t using breakmarriage operations. Hence, the total number of proposals is $O(n^2)$. Because of the computation of regret in each iteration, we might end up with a $O(n^3)$ time complexity. But Gusfield [3] explains how to use linked lists to represent regrets of women. It can be initiated in $O(n)$ time and takes $O(n)$ space. Using this structure, we no longer have to compute all values of regret and can reach a $O(n^2)$ complexity.

3.2 "Optimal" Stable Marriage

A slightly different approach is used by Irving, Leather and Gusfield [6] to search for an *"Optimal" Stable Matching*. Gale-Shapley algorithm can find a matching that favors the men/women at the expense of women/men. With a goal to find an "optimal" stable matching, Irving is maximizing the total sum of "satisfaction" of all men and women.

The satisfaction is measured by the position of a person's partner on his preference list. Egalitarian measure of optimality was first shown by McVitie and Wilson [39], they defined ranking as follows:

Definition 20 (Rank [39])**.** Let I be an instance of a Stable Marriage problem. Let Q be a stable matching in I. Let m be a man in I and w a woman in I. Then $\mathrm{r}(m, w)$ is the position of a woman w in the man m's preference list and $\mathrm{r}(w, m)$ is the position of a man m in the woman w's

preference list. The *value* of Q is defined as:

$$c(Q) = \sum_{i=1}^{n} \mathrm{r}(m_i, Q(m_i)) + \sum_{i=1}^{n} \mathrm{r}(Q(m_i), m_i)$$

We say that a stable matching Q is optimal if it has minimum possible value of $c(Q)$.

Clearly, a stable matching that minimizes this criterion maximizes the total satisfaction of all men and women in this instance. Irving gives an $O(n^4)$-time algorithm that finds an optimal stable matching.

Rotation poset P is defined as a directed graph structure to store *rotations* (Definition 11) in order of their *precedence* (Definition 14). The rotation poset is also defined as identical to our already known graph of rotations D (Definition 15). Even a sparse subgraph of rotation poset, denoted by P' is used here by Irving, which is again equivalent to a sparse subgraph G of D defined by Gusfield (definition 17). Therefore we will stick to our old naming convention for the sake of consistency.

Before explaining the algorithm to find an optimal stable matching, we need to define a *weight of a rotation.* Then we are going to show with one lemma and it's corollary that we can compute a value $c(Q)$ for any stable matching Q from a man optimal stable matching Q_0 and a subset of rotations that are on path from Q_0 to Q in G.

Definition 21 (Weight of rotation [6])**.** Let $m_0, \ldots, m_{k-1}$ be a subset of men, $w_0, \ldots, w_{k-1}$ be a subset of women and $\pi = \{(m_0, w_0), \ldots, (m_{k-1}, w_{k-1})\}$ be a rotation in an instance of Stable Marriage problem. Then the weight of the rotation π is defined as:

$$w(\pi) = \sum_{i=0}^{k-1} (\mathrm{r}(m_i, w_i) - \mathrm{r}(m_i, w_{i+1})) + \sum_{i=0}^{k-1} (\mathrm{r}(w_i, m_i) - \mathrm{r}(w_i, m_{i-1}))$$

($i \pm 1$ taken mod k)

Lemma 14 ([6])**.** Let Q be a stable matching in an instance of a Stable Marriage problem. Let π be a rotation exposed in Q, and let Q' be the stable matching obtained from Q by eliminating π. Then

$$c(Q') = c(Q) - w(\pi).$$

Proof. We refer to Irving, Leather and Gusfield [6]. □

Corollary 6 ([6])**.** Let Q be a stable matching in an instance of a Stable Marriage problem. Let $\pi_1, \ldots, \pi_t$ be a sequence of rotations that have to be eliminated in men optimal stable matching (in the order of succession) to receive the matching Q. Then

$$c(Q) = c(Q_0) - \sum_{i=1}^{t} w(\pi_i).$$

3.2.1 Finding the Maximum Weight Closed Subset of G

A network flow is used to find the maximum-weight closed subset of G (G being the sparse subgraph of rotation poset as defined in Definition 17). Given a graph G, the following capacitated $s-t$ flow graph $G(s,t)$ is defined.

Source vertex s and sink vertex t are added to the graph G. A directed edge is added to every "*negative vertex*" (every vertex π_i, where $w(\pi_i < 0)$. The capacity of every edge (s, π_i) is set to $|w(\pi_i)|$.

We add a directed edge from every "*positive vertex*" (every vertex π_j, where $w(\pi_j > 0)$) to vertex t is also added. These edges (π_j, t) have a capacity $w(\pi_j)$. Every original edge in G has the capacity set to infinity. [6]

Theorem 10 ([6])**.** Let X be the set of edges crossing a minimum $s-t$ cut in $G(s,t)$, and denote the weight of X by $w(X)$. The positive vertices in the maximum-weight closed subset of G are exactly the positive vertices whose edges into t are uncut by X. These vertices, and all vertices that reach them in G (predecessors), define a maximum-weight closed subset in G.

Proof. We denote the sets of positive and negative vertices in G by V^+ and V^- respectively. Let $N(W)$ be the set of all negative predecessors of vertices in any subset $W \subseteq V^+$.

Any negative vertex in a maximum-weight closed subset $C \subseteq G$ must precede at least one positive vertex in C, hence a maximum-weight closed subset of G can be defined as a subset

$$W = \arg \max_{W \subseteq V^+} w(W) - |w(N(W))|.$$

But then the problem of finding a maximum weight closed subset can be also interpreted as

$$W = \arg \min_{W \subseteq V^+} w(V^+ \setminus W) + |w(N(W))|.$$

Now let W be an arbitrary subset of V^+, and consider graph $G(s,t)$. If every edge from s to a vertex in $N(W)$ is cut, and every edge from a vertex in $V^+ \setminus W$ to t is also cut, then all paths from s to t are cut. Hence $w(X) \leq w(V^+ \setminus W) + |w(N(W))|$ for any $W \subseteq V^+$. Conversely, if we let $W^* \subseteq V^+$ consist of positive vertices whose edges to t are uncut by X, then, by definition, X cuts all edges to t from vertices in $V^+ \setminus W^*$, and X must certainly cut all the edges from s to vertices in $N(W^*)$, since X is an $s-t$ cut of finite capacity, and all original edges in G have infinite capacity. Hence

$$w(X) = w(V^+ \setminus W^*) + |w(N(W^*))| \leq w(V^+ \setminus W) + |w(N(W))|$$

for any arbitrary $W \subseteq V^+$. □

Theorem 11 ([6])**.** The maximum flow and minimum cut X in $G(s,t)$ can be found in $O(n^4)$ time.

Proof. When all capacities in $G(s,t)$ are integral, the running time of the Ford-Fulkerson algorithm is $O(EK)$, where K is the maximum $s-t$ flow value. All the capacities in $G(s,t)$ are integral, both E and K are $O(n^2)$, hence the theorem follows. □

Now we have an algorithm using a sparse directed graph G which represents the precedence of rotations, and with it, we can find the optimal stable matching using methods for finding the maximum flow, or a minimum cut, in $G(s,t)$. We also know that it takes $O(n^4)$ time to find it. Every step prior to that takes $O(n^2)$ time, as shown earlier.

3.3 Incremental Stable Marriage

In a real world, we can assume that our preferences are going to change. We have an old stable matching and new, modified preference lists. This problem is about searching for a stable matching with new preference lists that is the most similar (by the number of partner swaps) to the old stable matching.

Bredereck et al. [40] address adaptivity to changing environments by proposing "incrementalized version" of Stable Marriage problem. They ask a question: "What is the computational cost of adapting an existing stable matching after some of the preferences of the agents have changed." The problem is formally defined as:

Incremental Stable Marriage

Input:	Disjoint sets M and W of n agents each, two preference lists $\text{pref}_1, \text{pref}_2$ for two Stable Marriage problem instances $I_1 = (M, W, \text{pref}_1)$ and $I_2 = (M, W, \text{pref}_2)$, a stable matching for instance I_1 and a non-negative integer k.
Question:	Does I_2 admit a stable matching Q_2 such that $\text{dist}(Q_1, Q_2) = \lvert Q_1 \triangle Q_2 \rvert \leq k$? ($Q_1 \triangle Q_2$ denotes symmetric difference between sets Q_1 and Q_2)

We are given two preference lists, the old pref_1 and the new pref_2 for the same n men and n women. Our goal is to find a matching Q_2 within an instance of Stable Marriage problem with preference lists pref_2. This matching Q_2 must have a symmetric difference with the old matching Q_1 (that was created in an instance of stable matching I_1 using preference lists pref_1) lower or equal to some given threshold k. Therefore, we must try to look for the most similar stable matchings.

In a similar fashion to the previous section, we are going to use rotation poset – directed graph D (Definition 15) and it's sparse version – directed

graph G (Definition 17). Using a different function to determine weights of rotations we are once again reducing this problem to finding maximum weight closed subset of rotations.

To find a maximum weight subset, we obviously need a weight function $w(\pi)$. Let R_2 denote the set of all rotations for I_2. For every rotation $\pi \in R_2$, with $\pi = ((m_0, w_0), \ldots, (m_{r-1}, w_{r-1}))$ let

$$\begin{aligned} w(\pi) :=& |\{(m_i, w_{i+1}) \mid \{m_i, w_{i+1}\} \in Q_1, 0 \leq i \leq r-1\}| - \\ & |\{(m_i, w_i) \mid \{m_i, w_i\} \in Q_1, 0 \leq i \leq r-1\}| \end{aligned}$$

be the weight of rotation π. This weight function counts the number of pairs from Q_1 that the elimination of rotation π introduces minus the number of pairs from Q_1 that the elimination of rotation π removes. [40]

Again, similar to Optimal Stable Marriage's Lemma 14 and Corollary 6, we are now going to introduce a lemma and a corollary which allow us to measure a difference between two stable matchings by the weights of the rotations that have to be eliminated to reach from one stable matching to the other one. This value is obviously also dependent on Q_1.

Lemma 15 ([40])**.** Let Q_1 be a stable matching in the instance of Stable Marriage I_1. Let R_2 be a set of all rotations in the instance of Stable Marriage I_2 and let $\pi \in R_2$ be a rotation exposed in a stable matching Q for I_2, and let Q' be the stable matching obtained from Q by eliminating π. Then

$$|Q_1 \cap Q'| = |Q_1 \cap Q| + w(\pi).$$

Proof. In the following, all subscripts $i+1$ are taken modulo r. Let $\pi = ((m_0, w_0), \ldots, (m_{r-1}, w_{r-1}))$ be a rotation exposed in the stable matching Q, and let Q' be the stable matching obtained from Q by eliminating π.

$$\begin{aligned} |Q_1 \cap Q'| =& |Q_1 \cup (Q' \cup Q)| + |Q_1 \cup (M' \setminus M)| \\ =& |\{\{m, w\} \mid \{m, w\} \in Q \wedge (m, w) \notin \pi\}| + \\ & |\{\{m, w\} \mid \exists i,\ 0 \leq i \leq r-1,\ (m, w) = (m_i, w_{i+1})\}| \\ =& |\{\{m, w\} \mid \{m, w\} \in Q_1 \cup Q\}| - \\ & |\{\{m, w\} \mid \{m, w\} \in Q_1 \cup Q \wedge (m, w) \in \pi\}| + \\ & |\{\{m, w\} \mid \exists i,\ 0 \leq i \leq r-1,\ (m, w) = (m_i, w_{i+1})\}| \\ =& |Q_1 \cap Q| + w(\pi) \end{aligned}$$

□

Corollary 7 ([40])**.** Let Q_1 be a stable matching in the instance of Stable Marriage I_1. Let $C \subseteq R_2$ be a closed subset of rotations associated with stable matching Q for I_2 and let Q_0 be a men optimal stable matching for Stable Marriage instance I_2. Then

$$|Q_1 \cap Q| = |Q_1 \cap Q_0| + \sum_{\pi \in C} w(\pi).$$

Using this corollary, we can reduce this problem to finding a maximum-weight closed subset of rotations, as shown in the next lemma. After that, we are going to show that the sum of weights of rotations is at most n, hence finding such a subset of rotations can be done efficiently.

Lemma 16 ([40])**.** Let Q_1 be a stable matching in the instance of Stable Marriage I_1. Let Q_0 be man optimal stable matching for instance I_2 and Q_2 a stable matching for I_2. Let C be the closed subset of rotations associated to Q_2. Then, $\text{dist}(Q_1, Q_2) \leq k$ if and only if $\sum_{\pi \in C} w(\pi) \geq (\text{dist}(Q_1, Q_0) - k)/2$.

Proof. From the definition of symmetric difference, we know that

$$\text{dist}(Q_1, Q_2) = |Q_1| + |Q_2| - 2|Q_1 \cap Q_2|.$$

As we know, every stable matching in a given instance always matches the same set of agents, therefore $|Q_0| = |Q_2|$. Using this fact, and Corollary 7, we obtain

$$\begin{aligned} \text{dist}(Q_1, Q_2) &= |Q_1| + |Q_0| - 2(|Q_1 \cap Q_0| + \sum_{\pi \in C} w(\pi)) \\ &= \text{dist}(Q_1, Q_0) - 2 \sum_{\pi \in C} w(\pi) \end{aligned}$$

□

Lemma 17 ([40])**.** $\sum_{\pi \in R_2} w(\pi) \leq |Q_1|$ and finding a closed subset of rotations with maximum weight can be done in $O(n^3)$ time.

Proof. First, we need to create a definition of an intersection with rotation. Let $\pi = ((m_0, w_0), \ldots, (m_{r-1}, w_{r-1}))$ be a rotation. We define

$$\pi \cap Q_1 := \{(m_i, w_{i+1}) \mid \{m_i, w_{i+1}\} \in Q_1 \wedge (m_i, w_{i+1}) \in \pi\}$$

(as usual $i + 1$ is taken modulo r). Using the fact that each pair is at most in one rotation, we get

$$\sum_{\pi \in R_2} w(\pi) \leq \sum_{\pi \in R_2} |\pi \cap Q_1| = |\bigcup_{\pi \in R_2} (\pi \cap Q_1)| \leq |Q_1|$$

In Section 3.2.1 we talked about how finding a maximum-weight closed subset of rotations can be reduced to finding a minimum $s - t$ cut in a flow network bounded by the sum of the weights of the rotations. The number of vertices and edges in this sparse directed graph is in $O(n^2)$ and the sum of weights is in $O(n)$ as shown earlier in this proof. This problem can be solved by Ford-Fulkerson algorithm in $O(|E| * w)$, where $|E|$ is the number of edges and w is the cost of the minimum $s - t$ cut. Hence the algorithm runs in $O(n^3)$. □

Chapter 4

Our contribution

The goal of this thesis is to explore properties of Temporal Stable Marriage problem. Ideally we want to prove either:

- that it is *NP-hard* to solve it by providing a polynomial reduction from another *NP-hard* problem, or
- that it belongs to P, by finding an algorithm solving Temporal Stable Marriage in polynomial time.

At first, we are going to formally introduce the Temporal Stable Marriage problem and then we are going to show some of it's properties we have found. Towards the end, we discuss the approaches we tried out, but they haven't got us closer towards a solution and reasons why they have failed.

4.1 Temporal Stable Marriage

In real world scenarios, the preferences of people are constantly changing. Solution of classical Stable Marriage problem might not be stable after some of the participants change their priorities. If these changes are incremental, then we can use the algorithm for the Incremental Stable Marriage problem we introduced in Section 3.3.

But, if we know about these upcoming changes in advance, or we can guess them, and get all ℓ versions of preference lists of all participants, then the incremental method of modifying the matching with every change might not be the optimal solution. The problem can be even harder, when we try to find a single matching minimizing/maximizing some given metric over all ℓ instances of the Stable Marriage problem.

In our interpretation of this problem, we decided to work with the latter interpretation. As our metric, we chose *the maximum blocking pairs in an instance* over all ℓ given instances of Stable Marriage. Formally, we define this problem as:

Temporal Stable Marriage	
Input:	A set of n men $M = \{m_1, \ldots, m_n\}$, a set of n women $W = \{w_1, \ldots, w_n\}$, ℓ instances of Stable Marriage problem $I_1, \ldots, I_\ell$, where $\forall i \in \{0, \ldots, \ell\}$, $I_i = (M, W, \succ_i)$, and a positive integer k.
Question:	Is there a matching Q such that the maximum number of blocking pairs in every instance I_i is at most k.

4.2 Temporal Stable Marriage is NP-complete

In this section, we are going to talk about one of our results, which is NP-completeness of Temporal Stable Marriage even with $k = 2$. We achieve this result by a *polynomial reduction* from the 3-SAT problem.

The main idea is to create one instance of Stable Marriage for each clause from the 3-SAT problem, to simulate the logical AND relation between clauses, and to use parameter k as a "counter". This counter is used to simulate the logical OR relation between literals in clauses, by counting the number of literals in a clause that are not satisfied. Hence it is making sure that the clause, for which the given stable marriage instance was constructed, is satisfied. We define our reduction in Theorem 12 and prove that it is a correct polynomial reduction. At last, in Theorem 13, we complete the proof of *NP-completeness*.

Theorem 12. The Temporal Stable Marriage problem is NP-hard even if $k = 2$.

Proof. Let Φ be an instance of 3-SAT problem consisting of ℓ three-literal clauses $\{C_1, \ldots, C_\ell\}$ and n total variables. We define instance $\mathcal{I}_\Phi$ of Temporal Stable Marriage problem as follows:

For every clause $C_j \in \Phi$, where $j \in [1, \ell]$, we create one instance of stable marriage and label it I_j. For every variable x in Φ we add two men $m_x, m_{\neg x}$ and two women $w_x, w_{\neg x}$ to every instance of stable marriage $I_1, \ldots, I_\ell$. Preference lists of men and women in every I_j are formed as follows:

For every variable x that is not present in the clause C_j, we create preference lists of $m_x, m_{\neg x}, w_x, w_{\neg x}$ in I_j as follows:

$$m_x : w_x, w_{\neg x}, [\text{other women in arbitrary order}]$$
$$m_{\neg x} : w_{\neg x}, w_x, [\text{other women in arbitrary order}]$$
$$w_x : m_{\neg x}, m_x, [\text{other men in arbitrary order}]$$
$$w_{\neg x} : m_x, m_{\neg x}, [\text{other men in arbitrary order}]$$

Now we need to split the literals that appear in the clause C_j to positive literals, and negative literals. For the positive literals, we assign preferences as:

$$
\begin{aligned}
m_x &: w_x, w_{\neg x}, [\text{other women in arbitrary order}] \\
m_{\neg x} &: w_x, w_{\neg x}, [\text{other women in arbitrary order}] \\
w_x &: m_x, m_{\neg x}, [\text{other men in arbitrary order}] \\
w_{\neg x} &: m_x, m_{\neg x}, [\text{other men in arbitrary order}]
\end{aligned}
$$

At last, we define the preferences for negative literals as:

$$
\begin{aligned}
m_x &: w_{\neg x}, w_x, [\text{other women in arbitrary order}] \\
m_{\neg x} &: w_{\neg x}, w_x, [\text{other women in arbitrary order}] \\
w_x &: m_x, m_{\neg x}, [\text{other men in arbitrary order}] \\
w_{\neg x} &: m_x, m_{\neg x}, [\text{other men in arbitrary order}]
\end{aligned}
$$

It is clear that the for every:

- variable x not present in the clause $C_j \in \Phi$, the stable pairs in I_j are $\{(m_x, w_x),, (m_{\neg x}, w_{\neg x}), (m_x, w_{\neg x}), (m_{\neg x}, w_x)\}$ for each variable,
- positive literal x present in the clause $C_j \in \Phi$, each man and woman is in only one stable pair, and those are $\{(m_x, w_x), (m_{\neg x}, w_{\neg x})\}$, as (m_x, w_x) blocks the other two options,
- negative literal $\neg x$ present in the clause $C_j \in \Phi$, each man and woman is in only one stable pair, and those are $\{(m_x, w_{\neg x}), (m_{\neg x}, w_x)\}$, as now the pair $(m_x, w_{\neg x})$ blocks the other two options.

Any other pair is always blocked by some of these stable pairs mentioned above. They cannot ever be stable, as all men and women aligned with the same variable prefer each other to other men and women aligned with other variables.

To finish the construction, we set $k = 2$. This completes our construction of Temporal Stable Marriage problem instance $\mathcal{I}_\Phi$ from 3-SAT problem instance Φ. We still need to describe how the existence of acceptable matchings for this constructed instance of Temporal Stable Marriage problem relates to satisfiability of 3-SAT problem instance Φ.

Matching Q is stable for an single instance I_j if and only if:

- all pairs consist of a man and woman both aligned with the same variable x,
- for every positive literal x present in the clause $C_j \in \Phi$, pairs $\{(m_x, w_x), (m_{\neg x}, w_{\neg x})\} \in Q$,

- for every negative literal x present in the clause $C_j \in \Phi$, pairs $\{(m_x, w_{\neg x}), (m_{\neg x}, w_x)\} \in Q$.

Now we need to prove that the existence of assignment α satisfying Φ means that a matching Q_α satisfying $\mathcal{I}_\Phi$ exists and has at most k blocking pairs over all instances $I_1, \ldots, I_\ell$.

Let α be an assignment satisfying Φ. Let matching Q_α be currently empty. For each variable $x \in \alpha$ assigned the value *true*, we add pairs $\{(m_x, w_x), (m_{\neg x}, w_{\neg x})\}$ into matching Q_α. For each variable $x \in \alpha$ assigned the value *false*, we add pairs $\{(m_x, w_{\neg x}), (m_{\neg x}, w_x)\}$ into matching Q_α.

Now we need to prove that such a matching has actually at most $k = 2$ blocking pairs for each instance $I_j \in \mathcal{I}_\Phi$. Let I_j be an instance with strictly more than k blocking pairs for matching Q_α. We need to show that this would mean clause C_j is not satisfied in Φ, hence it cannot occur, as α is an assignment satisfying Φ. We know that the pairs in Q_α aligned with variables not present in C_j are not creating blocking pairs, as both sets of possible pairs are stable and are not creating blocking pairs (observed earlier in this proof). Hence, the only two options to create a blocking pair are:

- When C_j contains a positive literal x and $\{(m_x, w_{\neg x}), (m_{\neg x}, w_x)\}$, which, after checking the preferences, we know creates one blocking pair, or

- When C_j contains a negative literal x and $\{(m_x, w_x), (m_{\neg x}, w_{\neg x})\}$, which, after checking the preferences, we know creates one blocking pair.

We know, from how we defined matches in Q_α, that the truth value assigned to the variable x must not satisfy the respective literal in C_j. For each of these literals not satisfied, one blocking pair is added. But if there are strictly more than $k = 2$ blocking pairs, and we know that the clause C_j contains three literals, then all three literals are unsatisfied, hence the clause C_j is not satisfied. This shows us that the maximum number of blocking pairs in Q_α, cannot exceed $k = 2$ and, hence, is a valid solution.

When we proved this implication, we need to prove the opposite implication. To do that, we first need to prove, that the matching Q_α, acceptable for $\mathcal{I}_\Phi$, can never contain a pair(m_x, w_y) (respectively $(m_x, w_{\neg y})$) where $x \neq y$.

Let matching Q' contain a pair (m_x, w_y) (respectively $(m_x, w_{\neg y})$) with man and woman aligned with different literals x and y (resp. $\neg y$). The count of blocking pairs for I_j, where clause $C_j \in \Phi$ contains literal x, is incremented by at least three. This happens because pairs $(m_x, w_x), (m_x, w_{\neg x})$ are necessarily blocking, as m_x is the most preferred partner for both w_x and $w_{\neg x}$.

Finding the third pair is a little bit more complicated, as there is a multiple possible cases. In case the literal y (resp. $\neg y$) is also contained in the

clause $C_j \in \Phi$, then (m_y, w_y) (resp. $(m_{\neg y}, w_{\neg y})$) is necessarily a blocking pair, as w_y (resp. $w_{\neg y}$) is the most preferred woman for man m_y (resp. $m_{\neg y}$). This fact stays true in case the literal y (resp. $\neg y$) is not contained in the clause C_j.

The case when literal $\neg y$ (resp. y) is contained in the clause C_j is the most complicated, as w_y (resp. $w_{\neg y}$) is nobody's first choice. Here we must realize that one of the men $m \in \{m_y, m_{\neg y}\}$ must be either unpaired or paired with some $w \in \{w_z, w_{\neg z}\}$, where w is aligned with some different literal z ($z = x$ is possible). But this woman is surely less preferred than w_y (resp. $w_{\neg y}$), therefore the last blocking pair must be (m, w).

There can be multiple other blocking pairs in Q', but to show that it cannot be acceptable we need just three blocking pairs. Now we have shown that no acceptable matching for $\mathcal{I}_\Phi$ can contain a pair (m_x, w_y) (respectively $(m_x, w_{\neg y})$)

Finally, we can prove that if solution for $\mathcal{I}_\Phi$ exists, let it be matching Q_α, then assignment α satisfying Φ also exists.

Let α be an assignment. For each of it's variables $x \in \Phi$, we set the truth values as *true*, when Q_α contains pair (m_x, w_x), otherwise we set the truth values as *false* when Q_α contains pair $(m_x, w_{\neg x})$. We proved earlier, that one of these pairs have to be contained in Q_α, else it cannot be a valid solution for $\mathcal{I}_\Phi$ (it would imply that Q_α contains $(m_x, w_{\neg y}) \vee (m_x, w_y)$).

We claim that this assignment α satisfies Φ. Therefore, it must satisfy every clause $C_j \in \Phi$.

Let C_j be the clause not satisfied by the assignment α. This means that none of its three literals is satisfied.

We know that there is at most $k = 2$ blocking pairs for Q_α in I_j. From earlier observations we know, that the blocking pairs cannot come from pairs with agents aligned with variables not present in C_j. Hence there must be at least one variable of the three, let it be x, for which the blocking pair among it's adjacent agents does not exist.

That means that the matching Q_α necessarily contains (m_x, w_x), if C_j contains a positive literal of x, respectively contains $(m_x, w_{\neg x})$, if C_j contains a negative literal of x. But, from how we defined α earlier in this proof, this means that the literal of x is satisfied, hence C_j is satisfied, a contradiction. Therefore such assignment α satisfies Φ.

Now the Theorem 12 is proven, as existence of α implies existence of $\mathcal{I}_\Phi$, and existence of $\mathcal{I}_\Phi$ implies existence of α. □

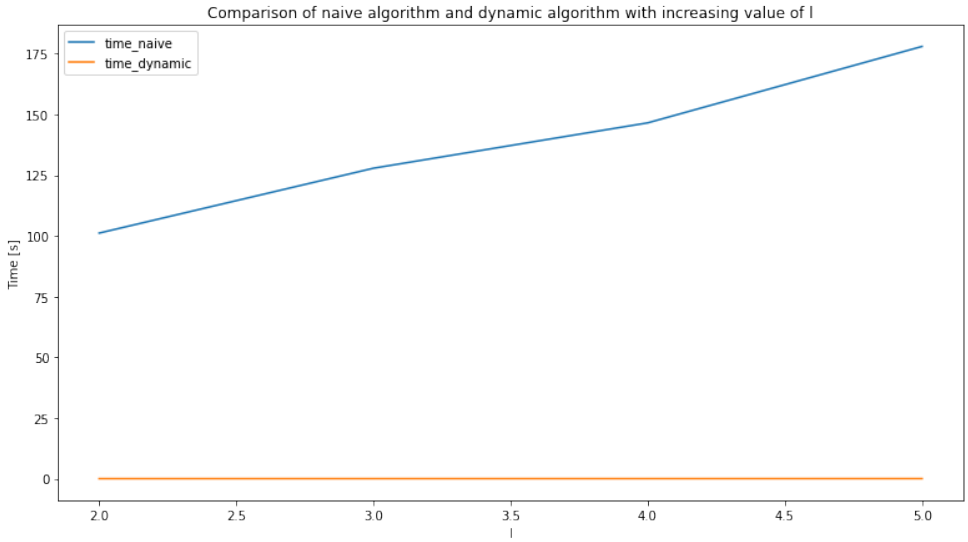

Figure 4.1: Speed comparison of naive brute force algorithm and algorithm utilizing dynamic programming. Blue is the naive algorithm, and orange is the dynamic algorithm, their speeds are relative to the count of stable marriage instances ℓ in the given instance of Temporal Stable Marriage problem

Theorem 13. The Temporal Stable Marriage problem is *NP-complete*.

Proof. From Theorem 12, we know that the Temporal Stable Marriage problem is *NP-hard*, now we just need to show that the solution can be verified in polynomial time. To verify a matching Q, we need to count the blocking pairs in each one of the ℓ instances of stable marriage problem in the Temporal Stable Marriage instance. To check one matching, we need up to $O(n^2)$ time, because each matching contains $O(n^2)$ edges. When we do this for all ℓ instances, we get $O(\ell n^2)$ time complexity. □

4.3 Algorithm Using Dynamic Programming

Naive algorithm is an option that is surely going to give us the correct solution to Temporary Stable Marriage problem, but it would have time complexity $O(n!)$. One of the most intuitive ways to speed up naive algorithm is by breaking down a larger problem into smaller subproblems – *dynamic programming* – solving each subproblem only once, and storing the solution in memory for future use. Compared to a naive algorithm, dynamic programming can significantly improve efficiency by avoiding repeated computations. A naive algorithm solves the same subproblem multiple times (e.g. counting

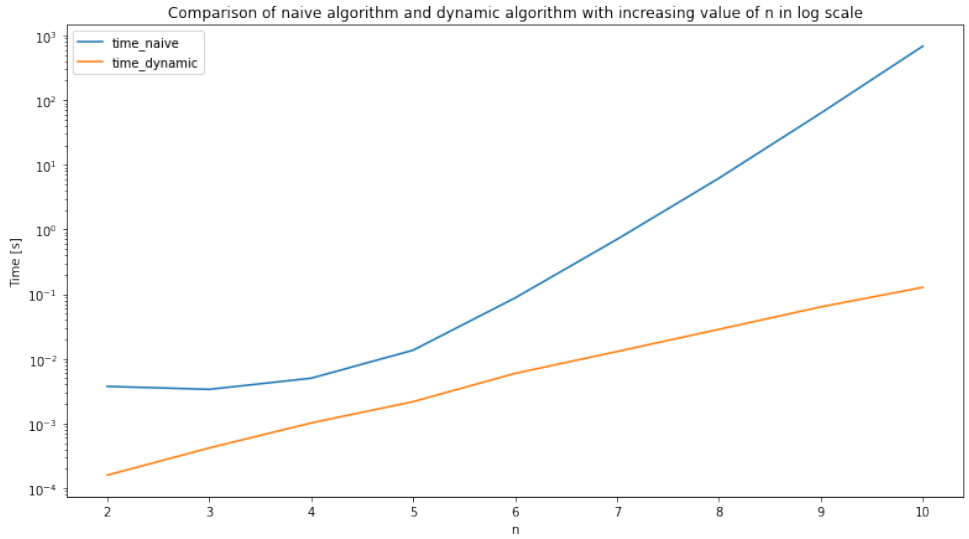

Figure 4.2: Speed comparison of naive brute force algorithm and algorithm utilizing dynamic programming in logarithmic scale of y axis. Blue is the naive algorithm, and orange is the dynamic algorithm, their speeds are relative to the size n of the given instance of Temporal Stable Marriage problem

the blocking pairs in the same subset of pairs), leading to an exponential increase in runtime as the size of the problem grows. But because of the inherent nature of blocking pairs, this algorithm only approximates the correct result for reasons given further below.

We utilize dynamic programming to break up our problem into smaller subinstances with j men and j women, $j \in \{1, \ldots, n\}$. We build our solution from bottom up, starting by choosing a partner for man m_1. With each of these possible pairs we create the first $j = 1$-th layer. Each $j + 1$-th layer of subinstances is constructed from the j-th layer by choosing a partner for man m_{j+1} and adding this pair. Partner of man m_{j+1} is chosen from the currently unpaired women.

Now if we follow this pattern, then in $j = n$-th layer, there will be all $n!$ solutions. The way how to reduce this number of solutions is for us to only save the best performing solution for each unique subset of men and women built while creating the layers.

This means that the size of each j-th layer is exactly $\binom{n}{j}$. It is caused by keeping only the solutions containing a distinct subset of women paired to men $\{m_1, \ldots, m_j\}$ in each layer.

We choose the best performing solution by evaluating the number of blocking pairs for each I_i, where $i \in \{1, \ell\}$, in each candidate. The candidate with the lowest maximum of blocking pairs in an instance is chosen. If none of them

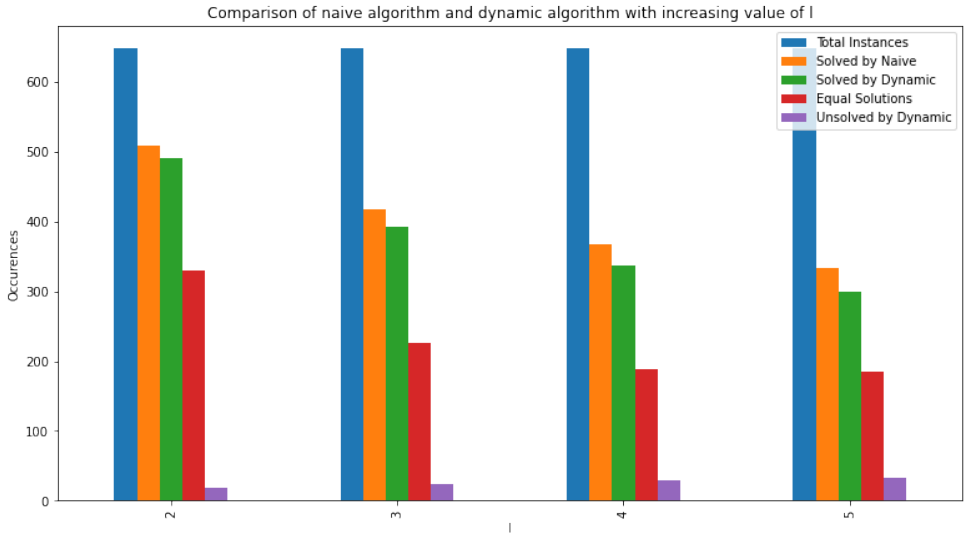

Figure 4.3: Comparison of solutions obtained from naive brute force algorithm and algorithm utilizing dynamic programming. Blue is the total amount of instances computed, orange is the amount of those instances for which the naive algorithm has found a solution, green is the amount of solutions found by the dynamic algorithm. Red is the count of times both algorithms have found a solution with the same amount of *max blocking pairs* and purple is the amount of instances, where naive algorithm found an acceptable solution, but dynamic algorithm haven't. All counts are grouped by the count of stable marriage instances ℓ in the given instance of Temporal Stable Marriage problem

has the maximum of blocking pairs in an instance lower than parameter k, then no solution is returned and saved to the new layer.

Pseudocode can be seen in Algorithm 4. We can see that the algorithm reuses the number of computed blocking pairs from previous iterations on line 10, therefore we only need to search for blocking pairs among the edges incidental with the newly added men m and woman w on line 9.

Another interesting section are lines 15 and 16. The algorithm must extract the maximum of blocking pairs in every run. This is because of the reusing of computed blocking pairs in next layers. If we stored just the max value, we would run into a problem where a different Stable Marriage instance overtake the previous maximum in the number of blocking pairs.

The matching, that is chosen to represent each of the combinations of men and women, is the first one of the matchings that have the minimal maximum of blocking pairs. It means that in later layers, this might create more than is the actual minimal maximum of blocking pairs for the given combination

Algorithm 4 Algorithm Using Dynamic Programming

```
1: function DynamicTemporalStableMarriage(n, ℓ, k, prefLists)
2:     leaves ← {{i} : {0}^ℓ | i ∈ {0, n − 1}}
3:     tree ← {1 : leaves}
4:     for depth in {1, ..., n − 1} do
5:         fullLayer ← {comb : {} | comb ∈ combinations(n, depth + 1)
6:         for vertex, oldBlocking in tree[depth] do
7:             for i in {0, n − 1} \ vertex do
8:                 newVertex ← vertex ∪ {i}
9:                 newBlocking ← newBlockingPairs(newVertex, prefLists)
10:                blocking ← newBlocking + oldBlocking
11:                fullLayer[sorted(newVertex)][newVertex] ← blocking
12:        newLayer ← {}
13:        for candidate in fullLayer.values() do
14:            blocksLists ← candidate.values()
15:            minBlocks ← min_{blocksList∈blocksLists} max_{b∈blocksList} b
16:            minMatching ← arg_{m∈candidate.keys()} minBlocks
17:            if max(minBlocks) ≤ k then
18:                layer[minMatching] ← minimum
19:        tree[depth + 1] ← layer
    return tree[n]
```

of men and women. Better way to choose the best performing solution might exist. Unfortunately we haven't found one that could find the optimal solution, without adding magnitude to time complexity.

Lemma 18. Algorithm Using Dynamic Programming for Temporal Stable Marriage problem creates $O(n2^n)$ candidate subsolutions in n layers. Each layer contains $O(\binom{n}{j})$ subsolutions, where j is the number of men in all of these subsolutions.

Proof. Let Q_j be a matching saved in j-th layer. By definition it contains j men paired with j women. We know that for each combination of women in set $\{Q_j(m_1), Q_j(m_2), \ldots, Q_j(m_j)\}$ there is one matching saved in j-th layer. Women from whom we are choosing are n women from the original instance of Temporal Stable Marriage problem. Hence there is $\binom{n}{j}$ possible different combinations of sets of women. The total number of subsolutions $n2^n$ comes from sum of all these layers $\binom{n}{1}, \ldots, \binom{n}{n}$ and the candidate subsolutions evaluated for them. □

We know that our algorithm evaluates $O(n2^n)$ different candidates for subsolutions. Each time we only need to verify the pairs incidental with newly added agents for each one of the preference profiles, meaning an additional $O(\ell n)$ time.

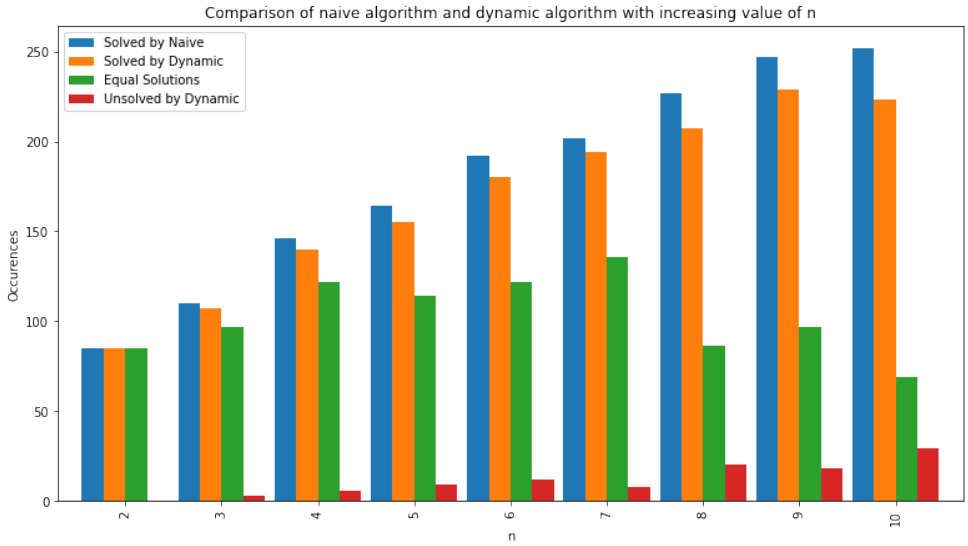

Figure 4.4: Comparison of solutions obtained from naive brute force algorithm and algorithm utilizing dynamic programming. Blue is the amount of instances for which the naive algorithm has found a solution, orange is the amount of solutions found by the dynamic algorithm. Green is the count of times both algorithms have found a solution with the same amount of *max blocking pairs* and red is the amount of instances, where naive algorithm found an acceptable solution, but dynamic algorithm haven't. All counts are grouped by the size n of the given instance of Temporal Stable Marriage problem

After computing blocking pairs for each of the candidates, we need to choose the one with minimal maximum of blocking pairs, which surely takes at most $O(\ell 2^n)$, as there can be at most 2^n candidates in a layer, and for each we need to check count of blocking pairs for each preference profile. As we check these candidates in every layer, and there are $n-1$ layers, evaluating candidates takes up to $O(\ell n 2^n)$ time.

If we combine all of these bits of knowledge, we end up with $O(\ell n^2 2^n)$ time complexity for Algorithm 4. Space complexity is $O(\ell n^2 2^n)$, as we need to store $O(n2^n)$ subsolutions of size $O(n)$ and for each subsolution we store ℓ counters of blocking pairs.

4.3.1 Comparison with naive brute force method

We tried to analyze the Algorithm 4 and compare it with a naive brute force method of searching for solution of a Temporal Stable Marriage problem instance. Our naive algorithm is sequentially trying out every possible matching for a given instance and counting the number of blocking pairs for

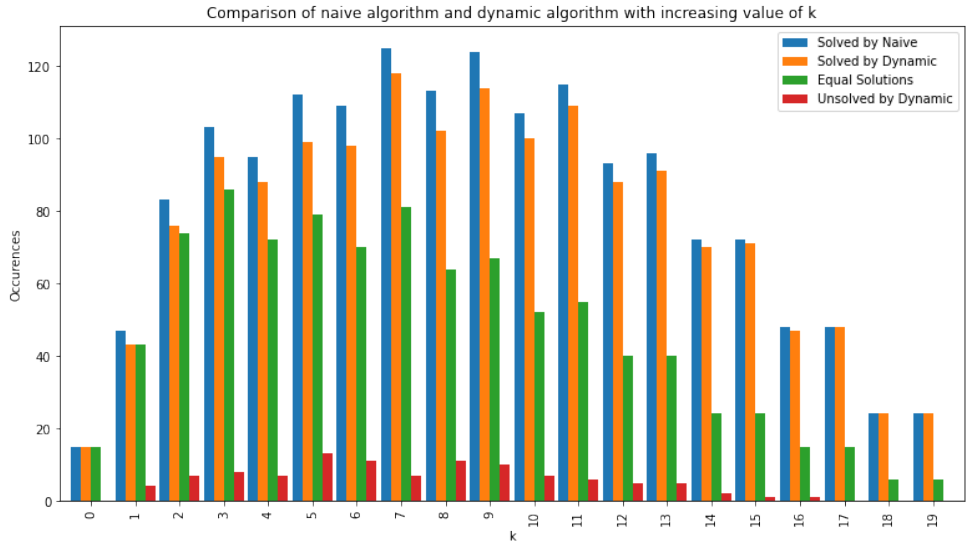

Figure 4.5: Comparison of solutions obtained from naive brute force algorithm and algorithm utilizing dynamic programming. Blue is the amount of instances for which the naive algorithm has found a solution, orange is the amount of solutions found by the dynamic algorithm. Green is the count of times both algorithms have found a solution with the same amount of *max blocking pairs* and red is the amount of instances, where naive algorithm found an acceptable solution, but dynamic algorithm haven't. All counts are grouped by the amount k of maximum acceptable *blocking pairs* in the given instance of Temporal Stable Marriage problem

all preference profiles (hence $O(ln^2n!)$ time complexity).

First, we have Figure 4.2 showing the comparison of the runtime of each algorithm in logarithmic scale. It is obvious that Algorithm 4's time complexity grows exponentially and the naive algorithm grows even faster. The speed difference is also huge, as Algorithm 4 finishes the computation for instances of size $n = 10$, on average, in $\sim 0.5s$, while the average computation time for naive algorithm is already at $\sim 1135s$.

In the next Figure 4.1, we can se that the difference made by the number of preference profiles in the given instance is just linear for both algorithms, as expected from their time complexities. Figures 4.3, 4.4 and 4.5 show us, most importantly, the amount of instances solved by the naive algorithm successfully, but not with Algorithm 4.

We can se that the number is rising in bigger instances both in terms of number of agents n and in terms of number of preference profiles ℓ. Out of all 105 instances, in which this occurred, from the total 2592 computed instances, only 26 instances had a possibility for a solution that had strictly less than k

max blocking pairs. Out of them, this best solution had 2 less max blocking pairs on 3 occasions, 3 less blocking pairs on 2 occasions and 1 less blocking pair in all others.

Chapter 5

Discussion

As there are numerous ways to grasp Temporal Stable Marriage, there were many approaches that we tried out but didn't receive any insight from, hence we evaluated them as "dead ends". Here we talk about our approaches to proving/disproving NP-hardness of Temporal Stable Marriage that didn't yield expected results. Most of them end with an unanswered question that prohibits us to continue that way. This could also mean that they are a few observations away from being useful.

5.1 Algorithm Repeating Incremental Stable Marriage

Our first naive attempt at creating an algorithm for Temporal Stable Marriage problem with $\ell = 2$. First we explain the fundamentals of the algorithm and in the end we describe why this algorithm doesn't work the way we intended.

Solution is computed by repeatedly calling the Incremental Stable Marriage algorithm with parameter $k_{ISM} = 4k$ (explained in Section 3.3) on every stable matching of I_1(enumerated by algorithm explained in Section 2.7.1). If the algorithm returns a matching, then we "merge" the two matchings into one and return as a solution.

At first we introduce an auxiliary definition of rotation on symmetric difference of two matchings, and it's properties. Then we use these definitions to simplify the explanation of our algorithm.

Definition 22 (Rotation on symmetric difference)**.** Let Q_1, Q_2 be two matchings in two instances of Stable Marriage problem with the same set of agents. Rotation on symmetric difference $\pi_{sd} = \{(m_1, w_1), \ldots, (m_{|Q_1 \setminus Q_2|}, w_{|Q_1 \setminus Q_2|})\}$ is a rotation of a subset of $Q_1 \setminus Q_2$ which, when eliminated, creates a subset of $Q_2 \setminus Q_1$ of the same size. We define maxRotation$(Q_1 \triangle Q_2)$ as the largest such rotation and $R(Q_1 \triangle Q_2)$ a set of all such rotations in $Q_1 \triangle Q_2$.

Lemma 19. Let Q_1, Q_2 be two matchings in two instances of Stable Marriage problem with the same set of agents. Then every edge of $Q_1 \setminus Q_2$ is part of exactly one rotation on symmetric difference.

Proof. From the definition of symmetric difference $Q_1 \triangle Q_2 = (Q_1 \cup Q_2) \setminus (Q_1 \cap Q_2)$ we already know that there cannot be an edge that is in both Q_1 and Q_2. Therefore the only option for the edges of symmetric difference is that for every $(m, w) \in Q_1 \setminus Q_2$ there exists $(m, w'), (m', w) \in Q_2 \setminus Q_1$, where $m \neq m'$ and $w \neq w'$, hence the proof is just a corollary of Theorem 5. □

Algorithm 5 Naive algorithm

```
1: function NaiveTemporal2(I1: instance of SM, I2: instance of SM, k: positive integer)
2:     for Q1 in EnumerateSM(I1) do
3:         Q2 ← IncrementalSM(I2, Q1, 2k)
4:         if |Q1△Q2| ≤ 4k and |maxRotation(Q1△Q2)| ≤ k then
5:             R1 ← {}
6:             R2 ← {}
7:             for π, in, sortedBySize(R(Q1△Q2)).desc() do
8:                 if |R1| < |R2| then
9:                     R1 ← R1 ∪ π
10:                else
11:                    R2 ← R2 ∪ π
12:            if R1 > k or R2 > k then
13:                continue
14:            Q ← Q1.eliminateRotations(R1)
15:            return Q
16:    return Null
```

With this lemma, we can now define the algorithm, which is described in Algorithm 5. During the computation, we enumerate stable matchings of one of the Stable Marriage instances I_1 in $O(n^2 + n|S|)$ time.

For each stable matching Q_1 in I_1 found, we try to find a similar matching Q_2, that is different in at most $2k$ pairs and none of the rotations found in the symmetric difference of Q_1 and Q_2 is bigger than k. If none is found, we try again for another Q_1.

Each computation of Incremental Stable Marriage to find a similar matching takes up to $O(n^3)$ time. As this is done for every enumerated matching Q_1, the whole cycle of searching for suitable Q_2 takes up to $O(n^4|S|)$ time.

After we find suitable Q_1 and Q_2, we can start with building the result Q. We split $R(Q_1 \triangle Q_2)$ into two sets of rotations of similar size. Both sets need to contain less than k pairs in their rotations, else we need to continue our

search for other suitable matchings. When we successfully find such sets of rotations, we construct the solution by taking Q_1 and eliminating all of the rotations contained in one of these sets.

For the splitting of $R(Q_1 \triangle Q_2)$, we need to sort the set of rotations by their sizes, which takes up to $O(k \log k)$ time. As these sets of rotations might be rejected by the algorithm, we need to take this time complexity into account, but we can assume that $k < n$ without a loss of generality (if $k \geq n$, then any matching is acceptable), and $O(k \log k) < O(n^3)$. Eliminating the rotations from Q_1 takes $O(k)$ time, but happens only once. This leaves us with the total time complexity of Algorithm Repeating Incremental Stable Marriage at $O(n^4|S|)$.

This algorithm gives us a matching Q that has both $|Q \triangle Q_2| < k$ and $|Q \triangle Q_1| < k$. The problem is that we have not found a way to "translate" the size of symmetric difference of two matchings into a count of blocking pairs, as each swap of partners can introduce up to $2n - 3$ blocking pairs, which is shown in the upcoming lemma.

Lemma 20. Let I be an instance of Stable Marriage problem with n men and women and Q a stable matching in I. Then swapping partners of two pairs $(m_i, w_i), (m_j, w_j)$ in Q might introduce $2n - 3$ blocking pairs.

Proof. Let I be an instance with preferences for all men m_i and women w_i, where $i \in \{1, \ldots, n\}$, as follows:

$$w_1 \succ_{m_i} w_2 \succ_{m_i} \cdots \succ_{m_i} w_n$$
$$m_1 \succ_{w_i} m_2 \succ_{w_i} \cdots \succ_{w_i} m_n.$$

Let $Q = \{(m_1, w_1), \ldots, (m_n, w_n)\}$ be a matching in I. We can check that Q is stable.

Let's consider swapping partner of pairs (m_1, w_1) and (m_n, w_n) in Q from now on. Everyone's favorite partner is m_1, resp. w_1. These agents are now paired with their least favorite partner m_n, resp. w_n.

This means that m_1 and w_1 create a blocking pair with every other possible partner, adding up to $2n-3$ pairs (n is the size of their preference lists, minus their current matchings and counting matching (m_1, w_1) just once). □

This means that we cannot efficiently use symmetric difference to help us with minimizing the number of blocking pairs. And because of that we cannot use this algorithm to solve Temporal Stable Marriage.

5.2 NP-hardness of Temporal Stable Marriage with Constant Number of Layers

We tried to reduce a chosen *NP-problem* to Temporal Stable Marriage problem with constant value of ℓ (we were trying $\ell \in \{2, 3\}$). When we did

that, we often encountered a problem of "simulating" both logical AND and logical OR relations while using a constant number of instances to do so for any sizes of sets of men and women.

One problem we faced was our inability to find a way to create an environment, where there are two sets of men $M_1 = \{m_0, m_1, m_2\}$ and $M_2 = \{m_3, m_4, m_5\}$ and two sets of women $W_1 = \{w_0, w_1, w_2\}$ and $W_2 = \{w_3, w_4, w_5\}$. We need to find such an instance, where matchings:

$$Q_0 = \{(m_i, w_i) \mid i \in [0, 5]\}$$
$$Q_1 = \{(m_i, w_{i+3 \bmod 6}) \mid i \in [0, 5]\}$$

are stable, and all other matchings produce $j \in \mathbb{N}^+$ blocking pairs. We needed this construction for reduction from Exact Cover by 3-Sets problem [41], for example. We would utilize it to check which elements are in a given subset if the subset is chosen (signified by M_1 being matched with women in set W_1), or the subset not being selected (signified by M_1 being matched with women in set W_2). It is easy to see why we need all pairs to be "selected" or "unselected" at once.

The reason for this construction not working is simple, and it's because of how rotations work. Let Q_0, described above, be the man-optimal stable matching for some instance $I = (M_1 \cup M_2, W_1 \cup W_2, \succ)$ of Stable Marriage problem. If we try to change the pairs with three simple rotations $\pi_i = \{(m_i, w_i), (m_{i+3}, w_{i+3})\}$, where $i \in \{0, 1, 2\}$, no rotation precedes any other. Hence, any subset $S \subseteq \{\pi_i \mid i \in \{0, 1, 2\}\}$ of rotations eliminated in Q_0 creates a stable matching which we do not want.

This shows us that it is needed to create one big rotation that changes all required pairs at once. But because all rotations are a circular chain, there is no way to achieve Q_1 using a single rotation.

Very similar problem occurred when we were searching for a way to create an environment, where at least one (or more) of three men $m_i \in M_1$ are matched to women $w_i \in W_1$ with matching index $i \in \{0, 1, 2\}$ (simulating logical OR between the options), else a fixed amount $j \in \mathbb{N}^+$ of blocking pairs appears. This construction was needed for reduction from 3-SAT problem, for example.

This time, three independent rotations $\pi_i = \{(m_i, w_i), (m_{i+3}, w_{i+3})\}$, where $i \in \{0, 1, 2\}$, create a problem, where by eliminating all $R = \{\pi_i \mid i \in \{0, 1, 2\}\}$ in Q_0 to create matching Q_1, it is also stable. And again, no bigger rotation can rotate these agents correctly. There is also an option to make (m_i, w_i), where $i \in [0, 5]$, the only stable pairs, but that means that any matching Q containing any pair (m_i, w_{i+3}) is not stable. We can still use the counting of blocking pairs and parameter k of Temporal Stable Matching instance to invalidate the matching if the number of blocking pairs gets above some threshold (we actually use this method in our proof of *NP-hardness* of Temporal Stable Marriage with unlimited number of instances ℓ in

Section 4.2). But with multiple of these constructions in one instance, the variability of interpretations for the number of blocking pairs (in an instance with three of these constructions, three blocking pairs could mean that each construction has one blocking pair, which would be acceptable, but it could also mean that one ot these constructions has three blocking pairs, which in this example should be rejected) would block it from any usefulness.

These properties of rotations stopped us from construction a proper polynomial reduction from *NP-complete* problems we tried out. There is still a possibility that a different utilization of multiple available instances of Stable Marriage at once, or the parameter k, might admit a reduction from *NP-complete* problem, but the techniques we tried were unsuccessful.

Conclusion

With the goal of tackling the Stable Marriage problem from the viewpoint of dynamically changing world, we have researched various available literature about Stable Marriage problem, it's modifications and similar problems. We mostly focused on the subset of literature, where a rotation poset is used, as we first expected our problem to be similar to Incremental Stable Marriage [40], where this construction plays a major role. Though we weren't able to utilize rotation poset much, as blocking pairs are not much similar to swaps or symmetric difference in usage, as shown in Lemma 20.

Our main contribution in this thesis is the definition of Temporal Stable Marriage problem and the proof of it's *NP-completeness* even with constant number of allowed blocking pairs in each instance, by polynomial reduction from 3-SAT problem. We also provided an algorithm utilizing dynamic programming that can approximate the matching with the lowest maximum of blocking pairs.

We also discussed the reasons why our planned utilization of rotation poset didn't work in our proposed algorithm naively repeating algorithm solving Incremental Stable Marriage. At the end we mentioned some of our most frequent reasons why we were unsuccessful in proving *NP-completeness* of Temporal Stable Marriage problem with small constant parameter ℓ.

There are some open questions after this thesis. One of them is the *NP-completeness* of Temporal Stable Marriage problem with small constant parameter ℓ. More general open question would be if there exists a right viewpoint from which we can better utilize blocking pairs, as this could pave the road for future research. It might be also needed to be able to find some algorithm solving the problem in polynomial time for some small constant values of ℓ.

Bibliography

1. GALE, David; SHAPLEY, Lloyd S. College admissions and the stability of marriage. *The American Mathematical Monthly*. 1962, vol. 69, no. 1, pp. 9–15. Available from DOI: https://doi.org/10.2307/2312726.
2. KNOP, Dušan; SCHIERREICH, Šimon; VALLA, Tomáš. NI-ATH: Základní pojmy. 2022. Available also from: https://courses.fit.cvut.cz/MI-ATH/lectures/files/lecture01.pdf. File accesible after loging into CTU network.
3. GUSFIELD, Dan. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*. 1987, vol. 16, no. 1, pp. 111–128. Available from DOI: https://doi.org/10.1137/0216010.
4. MANLOVE, David F; IRVING, Robert W; IWAMA, Kazuo; MIYAZAKI, Shuichi; MORITA, Yasufumi. Hard variants of stable marriage. *Theoretical Computer Science*. 2002, vol. 276, no. 1-2, pp. 261–279. Available from DOI: https://doi.org/10.1016/S0304-3975(01)00206-7.
5. ROTH, Alvin E.; VANDE VATE, John H. Random Paths to Stability in Two-Sided Matching. *Econometrica*. 1990, vol. 58, no. 6, pp. 1475–1480. ISSN 00129682, ISSN 14680262. Available from DOI: https://doi.org/10.2307/2938326.
6. IRVING, Robert W; LEATHER, Paul; GUSFIELD, Dan. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*. 1987, vol. 34, no. 3, pp. 532–543. Available from DOI: https://doi.org/10.1145/28869.28871.
7. IWAMA, Kazuo; MIYAZAKI, Shuichi. A survey of the stable marriage problem and its variants. In: *Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society*. USA: IEEE Computer Society, 2008, pp. 131–136. ICKS '08. Available from DOI: 10.1109/ICKS.2008.7.

8. *National Resident Matching Program* [https://www.nrmp.org/]. 2023. Accessed: April 25, 2023.

9. *Canadian Resident Matching Service* [https://www.carms.ca/]. 2023. Accessed: April 25, 2023.

10. *Japan Resident Matching Program* [https://www.jrmp.jp/]. 2023. Accessed: April 25, 2023.

11. ROTH, Alvin E.; SÖNMEZ, Tayfun; ÜNVER, M. Utku. Efficient Kidney Exchange: Coincidence of Wants in Markets with Compatibility-Based Preferences. *American Economic Review*. 2007, vol. 97, no. 3, pp. 828–851. Available from DOI: 10.1257/aer.97.3.828.

12. MOULIN, Hervé. *Handbook of Computational Social Choice*. Ed. by BRANDT, Felix; CONITZER, Vincent; ENDRISS, Ulle; LANG, Jérôme; PROCACCIA, Ariel D.Editors. Cambridge University Press, 2016. Available from DOI: 10.1017/CBO9781107446984.

13. ROTH, Alvin E.; SÖNMEZ, Tayfun; ÜNVER, M. Utku. Kidney Exchange. *The Quarterly Journal of Economics*. 2004, vol. 119, no. 2, pp. 457–488. ISSN 0033-5533. Available from DOI: 10.1162/0033553041382157.

14. ROTH, Alvin E.; SÖNMEZ, Tayfun; ÜNVER, M. Utku. Pairwise kidney exchange. *Journal of Economic Theory*. 2005, vol. 125, no. 2, pp. 151–188. ISSN 0022-0531. Available from DOI: https://doi.org/10.1016/j.jet.2005.04.004.

15. GUPTA, Sushmita; JAIN, Pallavi; ROY, Sanjukta; SAURABH, Saket; ZEHAVI, Meirav. On the (parameterized) complexity of almost stable marriage. In: SAXENA, Nitin; SIMON, Sunil (eds.). *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*. Germany: Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2020. Leibniz International Proceedings in Informatics, LIPIcs. Available from DOI: https://doi.org/10.4230/LIPIcs.FSTTCS.2020.24.

16. *Programs for Donor/Recipient Pairs with Incompatible Blood Types* [https://www.kidney.org/transplantation/livingdonors/incompatiblebloodtype]. 2019. Accessed: April 25, 2023.

17. MAGGS, Bruce M.; SITARAMAN, Ramesh K. Algorithmic Nuggets in Content Delivery. *Association for Computing Machinery Special Interest Group on Data Communication Computer Communication Review*. 2015, vol. 45, no. 3, pp. 52–66. ISSN 0146-4833. Available from DOI: 10.1145/2805789.2805800.

18. ROTH, Alvin E. The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory. *Journal of Political Economy*. 1984, vol. 92, no. 6, pp. 991–1016. Available from DOI: 10.1086/261272.

19. GALE, David; SOTOMAYOR, Marilda. Ms. Machiavelli and the Stable Matching Problem. *The American Mathematical Monthly*. 1985, vol. 92, no. 4, pp. 261–268. Available from DOI: 10.1080/00029890.1985.11971592.

20. ROTH, Alvin E. On the Allocation of Residents to Rural Hospitals: A General Property of Two-Sided Matching Markets. *Econometrica* [online]. 1986, vol. 54, no. 2, pp. 425–427 [visited on 2023-04-26]. ISSN 00129682, ISSN 14680262. Available from: http://www.jstor.org/stable/1913160.

21. IRVING, Robert W. Stable marriage and indifference. *Discrete Applied Mathematics*. 1994, vol. 48, no. 3, pp. 261–272. Available from DOI: https://doi.org/10.1016/0166-218X(92)00179-P.

22. KAVITHA, Telikepalli; MEHLHORN, Kurt; MICHAIL, Dimitrios; PALUCH, Katarzyna E. Strongly Stable Matchings in Time O(Nm) and Extension to the Hospitals-Residents Problem. *Association for Computing Machinery Transactions on Algorithms*. 2007, vol. 3, no. 2, 15–es. ISSN 1549-6325. Available from DOI: 10.1145/1240233.1240238.

23. IWAMA, Kazuo; MANLOVE, David; MIYAZAKI, Shuichi; MORITA, Yasufumi. Stable Marriage with Incomplete Lists and Ties. In: WIEDERMANN, Jirí; EMDE BOAS, Peter van; NIELSEN, Mogens (eds.). *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 443–452. ICALP '99. ISBN 978-3-540-48523-0. Available from DOI: https://doi.org/10.1007/978-0-387-30162-4_395.

24. GENC, Begum; SIALA, Mohamed; O'SULLIVAN, Barry; SIMONIN, Gilles. Robust Stable Marriage. *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 2017, vol. 31, no. 1. Available from DOI: 10.1609/aaai.v31i1.11107.

25. GENC, Begum; SIALA, Mohamed; O'SULLIVAN, Barry; SIMONIN, Gilles. Finding Robust Solutions to Stable Marriage. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2017. IJCAI '17. Available from DOI: 10.24963/ijcai.2017/88.

26. GENC, Begum; SIALA, Mohamed; SIMONIN, Gilles; O'SULLIVAN, Barry. On the Complexity of Robust Stable Marriage. In: GAO, Xiaofeng; DU, Hongwei; HAN, Meng (eds.). *Proceedings of the 11th International Conference on Combinatorial Optimization and Applications*. Cham: Springer International Publishing, 2017, pp. 441–448. COCOA '17. ISBN 978-3-319-71147-8. Available from DOI: https://doi.org/10.48550/arXiv.1709.06172.

27. GENC, Begum; SIALA, Mohamed; SIMONIN, Gilles; O'SULLIVAN, Barry. Complexity Study for the Robust Stable Marriage Problem. *Theoretical Computer Science*. 2019, vol. 775, pp. 76–92. ISSN 0304-3975. Available from DOI: https://doi.org/10.1016/j.tcs.2018.12.017.

28. KNUTH, Donald Ervin. *Marriage stables et leurs relations avec d'autres problèmes combinatoires*. Les Presses de l'université de Montréal, 1976.

29. IRVING, Robert W. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*. 1985, vol. 6, no. 4, pp. 577–595. ISSN 0196-6774. Available from DOI: https://doi.org/10.1016/0196-6774(85)90033-1.

30. BIRÓ, Péter; IRVING, Robert W.; MANLOVE, David F. Popular Matchings in the Marriage and Roommates Problems. In: CALAMONERI, Tiziana; DIAZ, Josep (eds.). *Algorithms and Complexity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 97–108. ISBN 978-3-642-13073-1. Available from DOI: https://doi.org/10.1007/978-3-642-13073-1_10.

31. HUANG, Chien-Chung; KAVITHA, Telikepalli. Popular matchings in the stable marriage problem. In: *38th International Colloquium on Automata, Languages and Programming, ICALP '11*. 2013, vol. 222, pp. 180–194. ISSN 0890-5401. Available from DOI: https://doi.org/10.1016/j.ic.2012.10.012.

32. KAVITHA, Telikepalli. A Size-Popularity Tradeoff in the Stable Marriage Problem. *SIAM Journal on Computing*. 2014, vol. 43, no. 1, pp. 52–71. Available from DOI: 10.1137/120902562.

33. IRVING, Robert W.; LEATHER, Paul. The Complexity of Counting Stable Marriages. *SIAM Journal on Computing*. 1986, vol. 15, no. 3, pp. 655–667. Available from DOI: 10.1137/0215048.

34. GUSFIELD, Dan; IRVING, Robert W. *The Stable marriage problem - structure and algorithms*. Cambridge, MA, USA: MIT Press, 1989. Foundations of computing series. ISBN 978-0-262-07118-5.

35. THURBER, Edward G. Concerning the maximum number of stable matchings in the stable marriage problem. *Discrete Mathematics*. 2002, vol. 248, no. 1, pp. 195–219. ISSN 0012-365X. Available from DOI: https://doi.org/10.1016/S0012-365X(01)00194-7.

36. STATHOPOULOS, Georgios K. *Variants of stable marriage algorithms, complexity and structural properties*. 2011. Available also from: http://mpla.math.uoa.gr/media/theses/msc/Stathopoulos_G.pdf. PhD thesis. University of Athens, Department of Mathematics–MPLA.

37. KARLIN, Anna R.; GHARAN, Shayan Oveis; WEBER, Robbie. A Simply Exponential Upper Bound on the Maximum Number of Stable Matchings. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. Los Angeles, CA, USA: Association for Computing Machinery, 2018, pp. 920–925. STOC '18. ISBN 9781450355599. Available from DOI: 10.1145/3188745.3188848.

38. PALMER, Cory; PÁLVÖLGYI, Dömotör. At most 3.55^n stable matchings. In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science*. 2022, pp. 217–227. FOCS '21. Available from DOI: 10.1109/FOCS52979.2021.00029.

39. MCVITIE, D. G.; WILSON, L. B. The Stable Marriage Problem. *Communications of the ACM*. 1971, vol. 14, no. 7, pp. 486–490. ISSN 0001-0782. Available from DOI: 10.1145/362619.362631.

40. BREDERECK, Robert; CHEN, Jiehua; KNOP, Dušan; LUO, Junjie; NIEDERMEIER, Rolf. Adapting Stable Matchings to Evolving Preferences. In: 2020, vol. 34, pp. 1830–1837. AAAI '20. Available from DOI: 10.1609/aaai.v34i02.5550.

41. GAREY, Michael R.; JOHNSON, David S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman and Co., 1990. ISBN 0716710455.