STRUCTURE AND COMPLEXITY OF THE STUDENT-PROJECT ALLOCATION PROBLEM

PEACE AYEGBA

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING

OCTOBER 30, 2025

Abstract

Matching problems occur in many practical settings where agents from one set need to be assigned to agents or resources in another. This thesis presents new results for a class of matching problems known as the *Student-Project Allocation problem* (SPA). In this problem, we are given a set of students, projects, and lecturers, where each project is offered by a single lecturer. Students have preferences over the projects they find acceptable, while lecturers may have no preferences, preferences over students, or preferences over projects. In the SPA model where both students and lecturers have preferences, the goal is to find a *stable matching*, which means an allocation of students to projects such that no student and lecturer would prefer an alternative assignment involving a different project. This thesis explores the complexity and structure of stable matchings in two variants of SPA.

In the Student-Project Allocation problem with lecturer preferences over Projects (SPA-P), stable matchings may vary in size, and the problem of finding a maximum-size stable matching (denoted MAX-SPA-P) is known to be NP-hard. Another variant is the Student-Project Allocation problem with lecturer preferences over Students, referred to as SPA-S. An extension of SPA-S where ties are allowed in the preference lists of both students and lecturers is known as SPA-ST. Similar to the SPA-P model, weakly stable matchings in SPA-ST may differ in size, and it is known that finding a maximum weakly stable matching (denoted MAX-SPA-ST) is NP-hard. In both MAX-SPA-P and MAX-SPA-ST, we examine how natural restrictions on the preference structure of students and lecturers affect the computational complexity of finding a maximum stable matching. We identify cases that admit polynomial-time algorithms and others that remain NP-hard. In addition, we study the parameterised complexity of MAX-SPA-P, and prove that the problem is fixed-parameter tractable with respect to a natural structural parameter.

Next, we consider the structural aspects of SPA-S. It is well known that a single instance may admit multiple stable matchings, and that the number of such matchings may grow exponentially with the input size. We present two new characterisations of the set of stable matchings for any given SPA-S instance. First, we prove that the set of stable matchings forms a distributive lattice under a natural dominance relation, in which the student-optimal and lecturer-optimal matchings correspond to the maximum and minimum elements, respectively. In the second characterisation, we extend the notion of *rotations*, originally defined for the one-to-one Stable Marriage problem, to the more complex SPA-S model. We introduce meta-rotations in SPA-S, and use this to develop the *meta-rotation poset*. We prove that there is a one-to-one correspondence between the stable matchings of a given SPA-S instance and the closed subsets of the associated meta-rotation poset.

Acknowledgements

First, I would like to thank Dr Sofiat Olaosebikan for giving me the opportunity to undertake this PhD under her supervision. I'm grateful for her guidance and for the care she takes in reading every draft of my work. I'm also thankful to my second supervisor Prof Kitty Meeks for her endless stream of brilliant ideas, for her encouragement and support during the more difficult times. A special thanks to Prof. David Manlove, my unofficial *third supervisor*, for his expertise in matching problems and his thoughtful feedback.

I am grateful to my examiners, Prof. Alice Miller and Dr. Christine Cheng, for their excellent feedback during my viva. I would also like to thank Dr. Jess Enright and Dr. Ciaran McCreesh for conducting my annual progression reviews and for their valuable writing suggestions. Thanks as well to the anonymous reviewers for their helpful comments on earlier versions of this work.

I've met many wonderful people at the university during the PhD, some of whom have become close friends: Nins, Laura, Kai, Newt, Doug, Ivaylo, Michael McKay, Fionnuala, Stephen, Alex, KayCee, and Lydia. Thank you for the fun conversations in the office, the coffee breaks, and the free therapy sessions. To my dearest friends: Debs, Ines, Lois, Tosan, Amy and Vic3 — thank you for helping me stay sane. A special thanks to Dr Bayo for his constant encouragements and for reading through parts my thesis draft. I can't name everyone, but if you ever asked "How's your PhD going?" or "When are you finishing?", you gave me the push I needed. Thank you!

To my Mum and Dad, thank you for the countless sacrifices that made this journey possible. To my siblings: Grace and Destiny, for always calling me at random just to check in, and *my only brother* Abraham, for especially for looking after me during the writing phase. To my best friend, Chibuike, thank you for always supporting me. You let me cry, made me laugh, and distracted me with football and movies. I'll always be grateful for your patience.

Lastly, and most importantly, to Abba: you've reminded me constantly this month that *the end of a thing is better than its beginning*. Thank you for guiding me throughout this journey and making sense out of my life. I know that my future is secure in your hands.

This PhD was supported by a College of Science and Engineering Scholarship from the University of Glasgow. Without that support, this would not have been possible.

Declaration

This thesis is submitted in accordance with the regulations for the degree of Doctor of Philosophy at the University of Glasgow. None of the material contained herein has been submitted for any other degree.

Parts of the results in Section 3.2.2 are based on ideas due to Prof David Manlove. The results in Section 3.4 are the product of collaboration with my second supervisor, Prof. Kitty Meeks. The work in Chapter 4 builds on results that first appeared in the thesis of my supervisor, Dr. Sofiat Olaosebikan, for a restricted version of the problem. All other results and contributions in this thesis are original and solely my own.

Publications

The following paper contains results that are presented in this thesis:

Peace Ayegba, Sofiat Olaosebikan, and David Manlove. Structural aspects of the Student-Project Allocation problem. Accepted for publication in *Discrete Applied Mathematics*. A preprint is available on arXiv at: arXiv:2501.18343.(This paper contains results presented in Chapter 4.)

Table of Contents

Acronyms

(1,TYPE)-SPA-P A restricted version of SPA-P where all students have identical preferences over projects. 48

COM-SMTI The problem of finding a complete weakly stable matching in a given SMTI instance. 21

COMPLETE SMTI-2ML The problem of finding a weakly stable matching of size at least n in an SMTI-2ML instance, where n is the number of men and women. 42, 43

HR Hospitals/Residents problem. 9, 10, 13, 28–31, 34, 35, 37, 76

HRC The Hospitals/Residents problem with Couples. 30

HRT Hospitals/Residents problem with Ties. 29, 30, 39

LCSM The Laminar Classified Stable Matching. 37

MAX SMTI-2ML-D The problem of finding a weakly stable matching of size at least k in an instance of SMTI-2ML. 42

MAX-HRT The problem of finding a maximum stable matching in HRT. 5, 7, 30, 39, 50, 73

MAX-SMTI The problem of finding a maximum stable matching in SMTI. 21, 35, 39, 50, 73

MAX-SPA-P The problem of finding a maximum stable matching in SPA-P. 9, 36, 39, 40, 45–48, 53, 72, 73

MAX-SPA-P-L1 A restricted version of SPA-P where there is only one lecturer. 47, 48, 72

MAX-SPA-P-SL The problem of finding a maximum stable matching in SPA-P-SL. 40, 47

MAX-SPA-PUC The problem of finding a maximum stable matching in SPA-PUC. 10, 53, 73

MAX-SPA-ST The problem of finding a maximum stable matching in SPA-ST. 9, 35, 39, 40, 42, 43, 72, 73

MIN MM The problem of finding a minimum maximal matching in a given bipartite graph. 46

SF Stable Fixtures problem. 14

SM Stable Marriage problem. 9, 10, 13, 15, 16, 20, 22, 23, 27, 28, 30

SMA Stable Multiple Activities problem. 14

SMI Stable Marriage problem with Incomplete lists. 19, 20, 23, 28, 29

SMT Stable Marriage problem with Ties. 20, 39

SMTI Stable Marriage problem with Ties and Incomplete lists. 20, 21, 28–30, 35, 39

SMTI-2ML SMTI where preferences are derived from a master list on both sides. 42

SPA Student–Project Allocation problem. 2, 8, 9, 12, 36

SPA-P Student–Project Allocation problem with lecturer preferences over Projects. 2, 6, 8–10, 36, 38–40, 45–48, 51–53, 72

SPA-P-SL A restriction of SPA-P where each student only finds projects from a single lecturer acceptable. 45, 47

SPA-PUC SPA-P with uniform capacities. 40, 51–54, 72, 73

SPA-S Student–Project Allocation problem with lecturer preferences over Students. 2, 8–11, 32, 34–37, 41, 45, 73, 74, 76

SPA-ST Student–Project Allocation problem with lecturer preferences over Students with Ties. 6, 9, 35, 36, 38–43, 72, 73

SR Stable Roommates problem. 13, 14

List of Figures

List of Tables

Chapter 1

Introduction

Combinatorial optimisation is a central topic in theoretical computer science, focusing on problems where the goal is to select an optimal solution from a finite, but potentially very large, set of feasible solutions. Matching problems form a quintessential class of such problems, with wide-spread real-world applications. Matching problems arise in settings where agents or resources must be assigned subject to capacity constraints and/or preferences. Examples include assigning access points to users in wireless networks [50], passengers to taxis in transportation systems [149], kidney donors to patients in hospitals [132], and students to projects in academic institutions [8]. In practical applications, the number of agents involved is often large, making manual assignment infeasible. For instance, the National Resident Matching Program in the United States [118] assigns more than 45,000 medical residents to hospitals each year. Since the outcome of these applications can have a direct impact on individuals' quality of life, it is essential that the algorithms used produce solutions that are perceived as fair, acceptable, and aligned with participants' expectations. This motivates both the design of efficient algorithms and the study of settings where such algorithms are unlikely to exist.

One important matching setting involves participants who express preferences over potential partners. These are known as matching problems involving preferences. At a high level, such a problem consists of a set of agents, each of whom specifies an ordering over a subset of the others based on their preferences. These preferences are typically ordinal, for example, an agent might list their first, second, third choice, and so on. In many settings, agents may also be subject to capacity constraints, meaning they can be assigned to only a limited number of partners. Matching problems involving preferences can be broadly classified into three categories: (i) bipartite matching problems with one-sided preferences, such as the Housing Allocation problem [5], where only one side expresses preferences over the other; (ii) bipartite matching problems with two-sided preferences, such as the Stable Marriage problem [54, 113], where both sides express preferences; and (iii) non-bipartite matching problems, such as the Stable Roommates problem [53], where each participant belongs to a single set and submits a preference list over all other participants. These categories and examples are discussed in detail in Chapter 2.

A solution to a matching problem is called a *matching*, which is an assignment of agents to acceptable partners that respects the given capacity constraints. In matching problems with preferences on both sides, it is often important to consider how well a matching satisfies the agents' preferences. However, beyond individual satisfaction, we also require the matching to be *stable*—that is, no subset of agents can form an alternative assignment among themselves in which every member is strictly better off than in the current matching. *Stability* ensures that once a matching is computed, no subset of participants has an incentive to deviate from it [129]. Its importance as a solution concept in matching problems where agents express preferences has been well established in the literature [129, 134].

In this thesis, we study the Student–Project Allocation problem (SPA), a class of matching problems involving three sets of agents: students, projects, and lecturers. In the SPA model with two-sided preferences, students have preferences over projects, each of which is offered by a lecturer who, depending on the variant considered, may have preferences over students, over projects, or over student–project pairs. The goal is to find a stable matching of students to projects that respects both project and lecturer capacity constraints. We examine both the algorithmic and structural aspects of two variants of SPA.

On the algorithmic side, we focus on the *Student-Project Allocation problem with lecturer preferences over Projects* (SPA-P), where both students and lecturers have preferences over projects. We present new complexity results by introducing natural restrictions on the input instance, and prove fixed-parameter tractability for selected NP-hard cases. On the structural side, we study the *Student-Project Allocation problem with lecturer preferences over Students* (SPA-S). We present two characterisations of the set of stable matchings admitted by any given SPA-S instance, highlighting the rich underlying structure of the problem. First, we show that the set of stable matchings forms a distributive lattice under a natural dominance relation. Second, we develop a partial order known as the meta-rotation poset and establish a one-to-one correspondence between the set of stable matchings in a given instance and the closed subsets of the poset.

1.1 Preliminaries

1.1.1 Complexity theory

Complexity theory is the study of how the amount of computational resource required to solve a problem, such as time or memory, grows with the size of the input. The input size, usually denoted by n, is a formal measure of the size of a problem instance, which in matching problems may include the number of agents involved and the lengths of their preference lists. An algorithm runs in *polynomial time* if its running time is bounded above by a polynomial function of the input size n, such as n, n^2, or n^3. Problems that can be solved by such algorithms are referred to as *tractable*. An algorithm runs in *polynomial time* if its running time is bounded above by n^k for

some constant k. Problems solvable by such algorithms are called *tractable*. In contrast, an algorithm runs in *exponential time* if its running time is bounded below by a^n for some constant $a > 1$; that is, its running time grows at least exponentially in the input size. Some problems even require *super exponential* time as those with running time $n!$. Problems that can only be solved by exponential-time or factorial-time algorithms are generally considered *intractable*, because the running time increases so rapidly that it becomes impractical to solve even moderately sized instances. Table 1.1 illustrates how logarithmic, polynomial, factorial, and exponential functions grow with input size n, highlighting the difference in their growth rates as n increases.

As the table illustrates, factorial and exponential functions grow far more rapidly than polynomial functions. For example, when $n = 50$, an algorithm with running time n^2 would require only 2,500 operations, which a modern computer could execute in just a few microseconds. In contrast, an algorithm with running time 2^n would require approximately 1.1×10^{15} operations, which would take about 13 days to complete assuming a computer performs 1 billion operations per second. Meanwhile, an algorithm with running time $n!$ would require approximately 3.0×10^{64} operations, taking more than 10^{47} years to finish.

n	$\log n$	n^2 (Polynomial)	2^n (Exponential)	$n!$ (Factorial)
5	2.32	25	32	120
10	3.32	100	1,024	3.6 million
20	4.32	400	$\approx 10^6$	$\approx 2.4 \times 10^{18}$
50	5.64	2,500	$\approx 1.1 \times 10^{15}$	$\approx 3.0 \times 10^{64}$

Table 1.1: Growth rates of logarithmic, polynomial, exponential, and factorial functions for increasing n

In practice, such growth rates may arise particularly for problems that involve evaluating a large number of feasible solutions. For example, certain instances of the stable matching problem admit exponentially many stable matchings [65, 94], making it infeasible to generate all stable matchings. Moreover, in practical settings, the goal is often not just to find a stable matching, but one that also satisfies an additional criterion, such as maximising the number of assigned agents. In such cases, a brute-force algorithm that enumerates all stable matchings to find an optimal one is impractical.

As a result, computational problems have been analysed and categorised formally, whereby these problems are classified into complexity classes based on their computational difficulty. These classifications typically consider *decision problems*, which are problems whose output for any given input is either YES or NO. The class P consists of all decision problems that can be solved in polynomial time. The class NP is the class of decision problems where, for every input for which the answer is yes, there exists a certificate (or proposed solution) that can be verified in

polynomial time by an algorithm. It is clear that $\mathsf{P} \subseteq \mathsf{NP}$, since any problem that can be solved in polynomial time can also be verified in polynomial time. However, it is widely believed that the converse does not hold, that is, $\mathsf{P} \neq \mathsf{NP}$. This suggests the existence of problems for which a given solution can be verified in polynomial time, but no polynomial-time algorithm exists for finding such a solution.

For example, given a graph G, we can verify in polynomial time whether the graph is connected, and we can efficiently find such a solution using a breadth-first or depth-first search algorithm.[1] Therefore, the graph connectivity problem is in the class P. On the other hand, consider the problem of *integer factorisation*, which asks for the non-trivial prime factors of a given integer N. If a factorisation of N is provided, it can be verified in polynomial time simply by multiplying the factors and checking that the product equals N. Thus integer factorisation belongs to the class NP. However, no polynomial-time algorithm is known for *finding* such a factorisation in general. Integer factorisation is of particular practical importance, since the presumed hardness of this task underlies widely used cryptographic algorithms [20, 58, 92].

To show that a problem is unlikely to be solvable in polynomial time, one typically proves that it is NP-hard. A problem is NP-hard if every problem in NP can be reduced to it in polynomial time. Thus, if a polynomial-time algorithm existed for any NP-hard problem, all problems in NP would also be solvable in polynomial time. A problem is NP-complete if it is both NP-hard and belongs to the class NP; such problems are considered the *hardest* problems in NP, in the sense that every problem in NP can be reduced to them in polynomial time. Assuming $\mathsf{P} \neq \mathsf{NP}$, no polynomial-time algorithm exists for any NP-complete problem. For further background on complexity classes and polynomial-time reductions, see [12, 45].

1.1.2 Coping with intractability

For computational problems that are intractable, several techniques can be used to obtain solutions that are useful in practice, even when no polynomial-time algorithms are known. In this thesis, we explore some of these approaches, including restricting aspects of the input to identify polynomial-time solvable cases, and applying tools from parameterised complexity theory and integer programming. These contributions are presented in Chapter 3. In this section, we briefly discuss these approaches as well as other techniques for coping with intractability, such as approximation algorithms and heuristic methods.

1.1.2.1 Approximation algorithms

An optimisation problem involves selecting the best solution from a set of feasible options, based on a given objective function and a set of constraints. An optimal solution is one that satisfies all

[1] A graph is *connected* if there is a path between every pair of vertices.

constraints and minimises or maximises the objective function. Since many interesting optimisation problems are NP-hard, a natural alternative is to find solutions that are close to optimal. Approximation algorithms run in polynomial time and produce solutions that are guaranteed to be close to the best possible. Their performance is measured by an approximation ratio, which compares the value t of the solution returned by the algorithm to the value OPT of an optimal solution, in the worst case. For minimization problems, the ratio is defined as $\frac{t}{\text{OPT}}$, and for maximization problems as $\frac{\text{OPT}}{t}$.

An algorithm is said to be a c-approximation algorithm if, for every input instance, the objective value of the solution it returns differs from the optimal objective value by at most a factor of c. More precisely, for minimization problems the algorithm always returns a solution whose objective value is at most c times the optimal value, and for maximization problems it always returns a solution whose objective value is at least $\frac{1}{c}$ times the optimal value. By relaxing the requirement to compute an exact optimal solution, approximation algorithms are a practical direction for solving problems that are unlikely to admit polynomial-time algorithms.

Approximation techniques have been applied to a range of NP-hard versions of matching problems. For example, in the Stable Marriage problem with Incomplete lists and Ties on one side (discussed in Section 2.1.2), Király [81] presented a linear-time $\frac{3}{2}$-approximation algorithm that returns a stable matching M with $|M| \geq \frac{2}{3} \times |M^*|$, where M^* is a stable matching of maximum size. That is, the algorithm guarantees a stable matching whose size is at least two-thirds of the largest possible. This result improves upon the previously best-known approximation factor of $\frac{5}{3}$, due to Irving and Manlove [67]. For a detailed discussion see the definitive textbook on approximation algorithms by Williamson and Shmoys, [146].

1.1.2.2 Heuristic methods

The aim of heuristic algorithms is to find feasible solutions quickly, often by following problem-specific rules or search strategies, rather than exhaustively exploring the entire solution space of a given problem. Although they do not guarantee optimal solutions or provide approximation bounds, they are often effective on large instances where exact methods are computationally infeasible. For instance, several heuristic algorithms have been proposed for a well-known matching problem, known as the Hospitals/Residents problem with Ties (see Section 2.2.2). In this setting, the problem of finding a weakly stable matching of maximum size, known as MAX-HRT, is NP-hard. Heuristic techniques for MAX-HRT have been proposed to either maximise the size of a weakly stable matching or minimise the number of blocking pairs. These include approaches based on greedy algorithms and local search strategies [21, 22, 115].

Similarly, Cao *et al.* [21] present a heuristic algorithm for MAX-HRT that constructs a stable matching incrementally by assigning residents to hospitals based on a scoring function, which removes the least-preferred resident whenever a hospital exceeds its capacity. In the context of

the Student–Project Allocation problem, Nguyen *et al.* [143] propose a heuristic for computing a maximum-sized weakly stable matching in the SPA-ST setting (see Section 2.3.2), where lecturers have preferences over students and ties may be present in the preference lists of both students and lecturers. A similar technique was subsequently developed for SPA-P (see Section 2.3.3) to find a stable matching of maximum size [145]. While heuristic methods lack worst-case guarantees, they perform well in practice and often scale effectively on large instances.

1.1.2.3 Fixed-Parameter Tractable algorithms

Fixed-parameter tractability offers a way to address NP-hard problems by restricting the exponential complexity of the problem to a selected parameter. Following the definition of Downey and Fellows [35], a problem is said to be *fixed-parameter tractable* (FPT) with respect to a parameter $k \in \mathbb{N}$ if it can be solved in time $O(f(k) \cdot n^c)$, where n is the input size, $c \in \mathbb{N}$ is a constant independent of k, and $f \colon \mathbb{N} \to \mathbb{N}$ is a computable function. This implies that while the running time may grow rapidly with the parameter k, it depends polynomially on the input size n.

FPT algorithms are particularly useful when the parameter is small in practice, even when the size of the input instance is large. However, not all NP-hard problems admit FPT algorithms with respect to a chosen parameter. To classify parameterised problems according to their complexity, Downey and Fellows [35] introduced the W-*hierarchy*, a sequence of complexity classes:

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP}.$$

Problems in $W[1]$ through $W[P]$ are widely believed not to admit fixed-parameter tractable algorithms. The conjecture that $\mathsf{FPT} \neq \mathsf{W}[1]$ is the central open question in parameterised complexity theory and is regarded as the analogue of the classical conjecture $\mathsf{P} \neq \mathsf{NP}$. The class XP contains problems that can be solved in time $O(n^{f(k)})$ for some computable function f. Since the parameter k appears in the exponent of the input size, even small increases in k can cause the running time to become extremely large, and problems of this form are not considered fixed-parameter tractable.

The class FPT generalises the classical complexity class P. Specifically, every problem in P is fixed-parameter tractable with respect to any choice of parameter; that is, it remains solvable in polynomial time regardless of how the input is parameterised. However, the inclusion is strict: the class FPT also contains parameterised versions of problems that are NP-complete in their classical form. A popular example is the *Vertex Cover* problem, which asks whether a given graph contains a vertex cover of size at most k. While this problem is NP-complete in general, it is fixed-parameter tractable (FPT) when parameterised by k. In particular, it admits algorithms with running time $O(2^k \cdot n)$ using bounded search trees or branching techniques [31, 107]. Moreover, more refined algorithms have achieved improved running times of $O(1.2738^k + kn)$ [25].

Although parameterised complexity has been extensively developed in the context of graph problems, its application to stable matching problems is still in its early stages. One known result in this direction is that MAX-HRT is FPT when parameterised by the size of the matching [9], or by the total length of ties in the instance [106]. Further discussion of parameterised complexity in the context of matching problems is provided in Section 3.4.1. Approximation techniques and parameterised complexity are often studied as separate approaches for handling intractable problems. However, recent work has shown that combining these strategies can yield more powerful algorithmic techniques.

Marx [105] pointed out that, for some problems, there are no known approximation algorithms and no fixed-parameter tractable algorithm with respect to any known parameter. However, such problems can sometimes be tackled using *parameterised approximation algorithms*, which provide approximate solutions within a running time of the form $f(k) \cdot n^c$, where k is a parameter and n is the input size. This approach combines ideas from approximation and parameterised complexity for problems where neither approximation nor fixed-parameter techniques are effective on their own.

1.1.2.4 Integer and Linear Programming

Many combinatorial optimisation problems can be expressed as either linear or integer linear programs. *Linear programming* (LP) involves optimising a linear objective function subject to linear inequality constraints, where variables can take any real values. Linear programs can be solved in polynomial time using algorithms like interior-point methods [77, 79, 142]. An *integer linear program* (ILP) is a linear program in which all variables are constrained to take integer values, making the model more expressive but also significantly harder to solve. The decision version of ILP asks whether there exists an integer solution that satisfies all given linear constraints and achieves at least a specified objective value. This problem is NP-complete and was included in Karp's list of 21 classical NP-complete problems [45, 78]. Mixed-integer linear programs (MILPs), where only a subset of variables must be integers, are even more general and typically harder to solve in the worst case.

Matching problems can also be modelled using ILPs. A common approach is to relax the integrality constraints to obtain an LP, which is generally easier to solve. In some cases, this LP relaxation remains *integral*, meaning that all optimal solutions are already integer-valued. When this occurs, the problem can be solved efficiently using LP methods alone. Several stable matching problems, including the Stable Marriage problem, exhibit this property [4, 80, 131]. For any SM instance of size n, it has been shown that we can efficiently construct a set of linear inequalities of polynomial size in n such that there is a one-to-one correspondence between the stable matchings of the instance and the extreme points of the polytope[2] defined by these inequalities [54, 144].

[2]In this context, the polytope is the set of all solutions (possibly fractional) that satisfy the matching and stability constraints. Each stable matching corresponds to a vertex (or extreme point) of this set.

With this representation, other stable matching problems, such as finding an egalitarian or minimum regret matching, can be solved using general LP methods, since an optimal solution will correspond to one of these extreme points.

Although ILPs are NP-complete, modern solvers such as CPLEX [1] and Gurobi [2] perform well in practice. These solvers have been successfully applied to real-world stable matching problems. For example, ILP formulations for NP-hard variants of the Stable Marriage and Hospital/Residents problems have been used to solve instances involving up to 50,000 agents on each side within seconds [34]. ILPs also play an important role in parameterised complexity. While solving ILPs is NP-complete in general, the problem is fixed-parameter tractable when parameterised by the number of variables [31]. This has led to the development of parameterised algorithms that model problems as ILPs in which the number of variables is bounded by a function of the parameter [46, 83, 114]. This technique allows ILP solvers to efficiently handle otherwise intractable problems [119].

1.2 Thesis Statement

The Student–Project Allocation problem (SPA) arises in many practical applications, and exhibits rich algorithmic and structural properties. In this thesis, we prove new complexity results for variants of SPA by imposing natural restrictions on the input instance. For some intractable cases, we show that by identifying and exploiting suitable structural parameters, the problem becomes fixed-parameter tractable with respect to those parameters. In addition, we develop new characterisations of the set of stable matchings, which enable efficient algorithms for computing stable matchings with desirable properties.

1.3 Contributions and Thesis Outline

In this thesis, we examine the algorithmic and structural aspects of well-known variants of the Student–Project Allocation problem (SPA). We begin by analysing the computational complexity of the Student-Project Allocation problem with lecturer preferences over Projects (SPA-P). We present both polynomial-time algorithms and NP-hardness results for finding a maximum-size stable matching in SPA-P under natural restrictions. For intractable cases, we introduce a parameterised version of SPA-P, where we introduce *project topics*. We prove that the problem is fixed-parameter tractable when parameterised by the number of project topics. This means that, although finding a maximum-size stable matching in SPA-P is NP-hard in general, it can be solved efficiently in practice when the number of project topics is small, since the running time grows quickly with the number of topics but remains polynomial in the size of the input.

In the second part of the thesis, we shift our focus to the Student-Project Allocation problem with lecturer preferences over Students(SPA-S). We first prove that the set of stable matchings forms

a distributive lattice, in which the student-optimal matching is the unique maximum element and the lecturer-optimal matching is the unique minimum element. We then extend the classical notion of *rotations*, originally developed for the Stable Marriage (SM) and Hospital Residents (HR) problems [14, 26, 52, 54], to the SPA-S setting. Building on this generalisation, we develop the *meta-rotation poset*, a compact structure that encodes the entire set of stable matchings in any given instance. Additionally, we prove that the set of stable matchings in a given instance is in one-to-one correspondence with the closed subsets of the poset. This structure supports efficient algorithms for identifying all stable pairs, enumerating all stable matchings, and provides new insights into the properties of the set of stable matchings in any given SPA-S instance.

The thesis is organised as follows:

- *Chapter 2: Literature Review.* This chapter surveys relevant work on classical stable matching problems such as the Stable Marriage problem (SM) and the Hospital/Residents problem (HR). Thereafter, we focus on the Student-Project Allocation problem (SPA) and its extensions including the *Student-Project Allocation problem with lecturer preferences over Projects* (SPA-P), *Student-Project Allocation problem with lecturer preferences over Students* (SPA-S), and *Student-Project Allocation problem with lecturer preferences over Students including Ties* (SPA-ST). We focus on known computational complexity results for these variants, structural characterisations, and algorithmic techniques that support the contributions made in this thesis.

- *Chapter 3: Complexity results for restricted* SPA *variants.* In this chapter, we examine how imposing natural restrictions on the input instance influences the computational complexity of two variants of SPA, namely SPA-P and SPA-ST. We begin with SPA-P, where both students and lecturers express preferences over projects. In this setting, stable matchings can vary in size, and the problem of finding a maximum-size weakly stable matching, denoted MAX-SPA-P, is known to be NP-hard [102]. We show that the MAX-SPA-P problem remains NP-hard even when the preferences of both students and lecturers are consistent with a single master list over projects. On the positive side, we show that if every student ranks only projects offered by a single lecturer, then a maximum-size stable matching can be computed in polynomial time.

  Then we consider the SPA-ST model, in which lecturers express preferences over students, and ties are permitted in the preference lists of both students and lecturers. In this setting, there are three notions of stability: weak, strong, and super-stability. Under weak stability, stable matchings can differ in size, and the problem of computing a maximum-size weakly stable matching (MAX-SPA-ST) is known to be NP-hard [73, 100]. Moreover, the problem is NP-hard even if the ties are present at the end of preference lists and on one side only, each tie is of length 2, and there is at most one tie per list [100]. We strengthen this result by

proving that the problem remains NP-hard even when the instance involves only a single lecturer.

Finally, we consider the parameterised complexity of SPA-P. We introduce a natural structural parameter, *project topics*, whereby each project is associated with a project topic and students express strict preferences over topics rather than over individual projects. We further assume that each lecturer has the same capacity as each of the projects that they offer, so that project and lecturer capacities are uniform. We denote this problem as MAX-SPA-PUC. We prove that MAX-SPA-PUC is fixed-parameter tractable when parameterised by the number of project topics, despite being NP-hard in the general case.

- *Chapter 4: Structural results for* SPA-S. In this chapter, we study the structure of the set of stable matchings in SPA-S, where students have preferences over projects and lecturers have preferences over students. It is well-known that the set of stable matchings in the Stable Marriage (SM) and Hospital/Residents (HR) problem forms a distributive lattice under a natural partial order [54]. Moreover, previous work has shown that a similar result holds for SPA-S under the restriction that each student has preferences only over projects offered by different lecturers [121]. We build substantially on this result by showing that the set of stable matchings forms a distributive lattice in the general case (without this restriction), whereby, students may express preferences over multiple projects offered by the same lecturer.

  We define the *meet* and *join* operations on pairs of stable matchings in SPA-S, where each student is assigned to the more preferred (meet) or less preferred (join) of their projects between two stable matchings. We show that applying either operation to any two stable matchings always yields another stable matching. Thereafter, we prove that the set of all stable matchings in a given SPA-S instance forms a distributive lattice under the partial order defined by student preferences. We also present additional structural results that arise specifically in SPA-S due to the possibility that students may find multiple projects offered by the same lecturer acceptable. These properties do not occur in classical models such as SM and HR, and they inform our definition of *rotations* and the *meta-rotation poset* which we develop in Chapter 5.

- *Chapter 5: Meta-rotation poset for* SPA-S. In this chapter, we introduce the notion of *meta-rotations*, which generalises the classical concept of rotations from the SM and HR models to the SPA-S context. We show that the set of stable matchings in any given instance of SPA-S is in one-to-one correspondence with the closed subsets of a partial order known as the meta-rotation poset. This *meta-rotation poset* provides a compact representation of the set of stable matchings in any SPA-S instance. We prove that the meta-rotation poset can be constructed in polynomial time and used to efficiently traverse the lattice of stable matchings. This structure has several key algorithmic consequences: it allows us to enumerate

all stable matchings, identify all stable pairs, and analyse the relationships between different stable matchings. In addition to these algorithmic applications, our results reveal new structural properties of the set of stable matchings in SPA-S, providing insight into the computational complexity of related optimisation problems.

Chapter 2

Literature Review

As mentioned earlier, matching problems that include preferences can be broadly classified into three main categories: bipartite matching with one-sided preferences, bipartite matching with two-sided preferences, and non-bipartite matching problems. These categories are summarized in Figure 2.1. In this chapter, we briefly review each of these categories and then focus on problems within the class of bipartite matching with two-sided preferences, which is the most relevant to the work presented in this thesis.

Specifically, in Section 2.1, we begin with the *Stable Marriage* problem (SM), presenting key results and extensions that are directly relevant to our work. In Section 2.2, we examine the *Hospitals/Residents* problem (HR) and its extensions, focusing on the key structural and algorithmic results established in the literature. In Section 2.3, we discuss the Student-Project Allocation problem (SPA) and provide a detailed overview of its main variants, highlighting significant algorithmic and structural results. Finally, in Section 2.4, we discuss some models that are closely related to SPA, drawing attention to their similarities and differences.

Bipartite matching problems with one-sided preference: In this setting, we have a set of agents and a set of indivisible resources (such as houses), where agents express preferences over resources. A well-known example is the *House Allocation* problem (HA) [3, 5, 6], where each agent is assigned at most one house based on their preferences. An extension of this model is the *Capacitated House Allocation* problem (CHA) [104], where each house can accommodate multiple agents. In both models, much of the literature focuses on finding matchings that are Pareto-optimal [5, 104] or Popular [30] [1], and efficient algorithms have been developed to compute them. More recently, Santhini *et al.* [136] introduced new notions of optimality for these models, including the concept of *weak dominance*, which addresses questions such as: "Does there exist an assignment that matches at least 50% of applicants to their top choice, and at least 75% to one of their first or second choices?" They also developed randomised algorithms to find such

[1] A matching is *Pareto optimal* if no other matching makes some agents strictly better off without making any agent worse off. A matching is *popular* if no other matching is preferred by a majority of agents.

solutions, particularly in instances with fixed quotas.

Bipartite matching problems with two-sided preferences: In this setting, agents are divided into two disjoint sets, and both sides rank members of the other. This category includes some of the most studied problems in matching theory, such as the *Stable Marriage* problem (SM) [54], the *Hospitals/Residents* problem (HR) [43], and the *Student–Project Allocation* problem (SPA) [7]. We review these problems in detail in Sections 2.1 to 2.3. Informally, the SM problem models a one-to-one matching problem in which each man expresses preferences over all women, and each woman similarly expresses preferences over all men. The goal is to compute a *stable matching*, in which every man is assigned to exactly one woman and vice versa, and there is no pair of a man and a woman who are not assigned together but would both prefer each other to their assigned partners.

The *Hospitals/Residents* problem (HR) generalises the Stable Marriage problem (SM) to a one-to-many setting. In this model, residents have preferences over hospitals, each hospital has preferences over residents, and each hospital also has a capacity indicating the maximum number of residents it can accept. The notion of stability is extended to this setting in a straightforward way. Further generalisations allow both sides to have capacities, leading to many-to-many matching models such as the *Stable Allocation* problem [15, 18, 33], where each agent may be matched to multiple partners. Several structural and algorithmic results from the one-to-one SM setting have been extended to these more general models [14, 36, 41, 90].

Non-bipartite matching problems with preferences: In non-bipartite matching problems, all participants belong to a single set and express preferences over one another. This differs from bipartite settings, where agents are partitioned into two disjoint sets, and each agent ranks only members of the opposite set. A well-known example of a matching problem in the non-bipartite setting is the *Stable Roommates* problem (SR) [48, 53, 62], where agents are paired based on mutual preferences. As in SM and HR, the goal is to compute a *stable matching*, defined as a matching in which no pair of agents would both prefer to be matched with each other rather than with their current partners. In this sense, SR can be viewed as a natural generalisation of the stable marriage model to a non-bipartite setting.

Unlike SM, where a stable matching is guaranteed to exist for every instance, the Stable Roommates problem (SR) has the key limitation that some instances admit no stable matching [54]. Nevertheless, when a stable matching does exist, it can be found using the polynomial-time algorithm of Irving [62]. Moreover, in such cases, all stable matchings include the same set of assigned agents and are of equal size [54]. For instances that do not admit any stable matching, Gusfield and Irving [54] raised the question of whether a succinct and verifiable certificate could be provided to prove the absence of a stable matching. This question was answered affirmatively by Tan [139,140], who introduced a combinatorial structure known as a *stable partition*. A stable

partition generalises a stable matching by dividing the set of agents into disjoint subsets, where each subset is either a singleton (representing an unassigned agent) or a cycle of length at least two. In each cycle, every agent prefers their assigned neighbours at least as much as any agent outside the cycle.

Every instance of SR, regardless of whether it admits a stable matching, is guaranteed to admit at least one stable partition [139]. Moreover, if a stable matching exists, it corresponds to a stable partition in which all cycles are of length two. As a result, recent research has explored the use of stable partitions to analyse the structural properties of SR instances [48, 49]. The non-bipartite setting also includes more general models such as the *Stable Fixtures* problem (SF) [49, 72, 94] and the *Stable Multiple Activities* problem (SMA) [23, 24], which are many-to-many generalisations of SR. In SF, each agent may be assigned to several others, up to a specified capacity, and has a strict preference list over a subset of the other agents. In the SMA model, a pair of agents may be assigned in many different ways, representing various forms of collaboration or joint activity, and agents may express preferences over both their partners and the form of assignment. Cechlárová and Fleiner [23] show that SMA can be reduced to an instance of SR with incomplete preference lists.

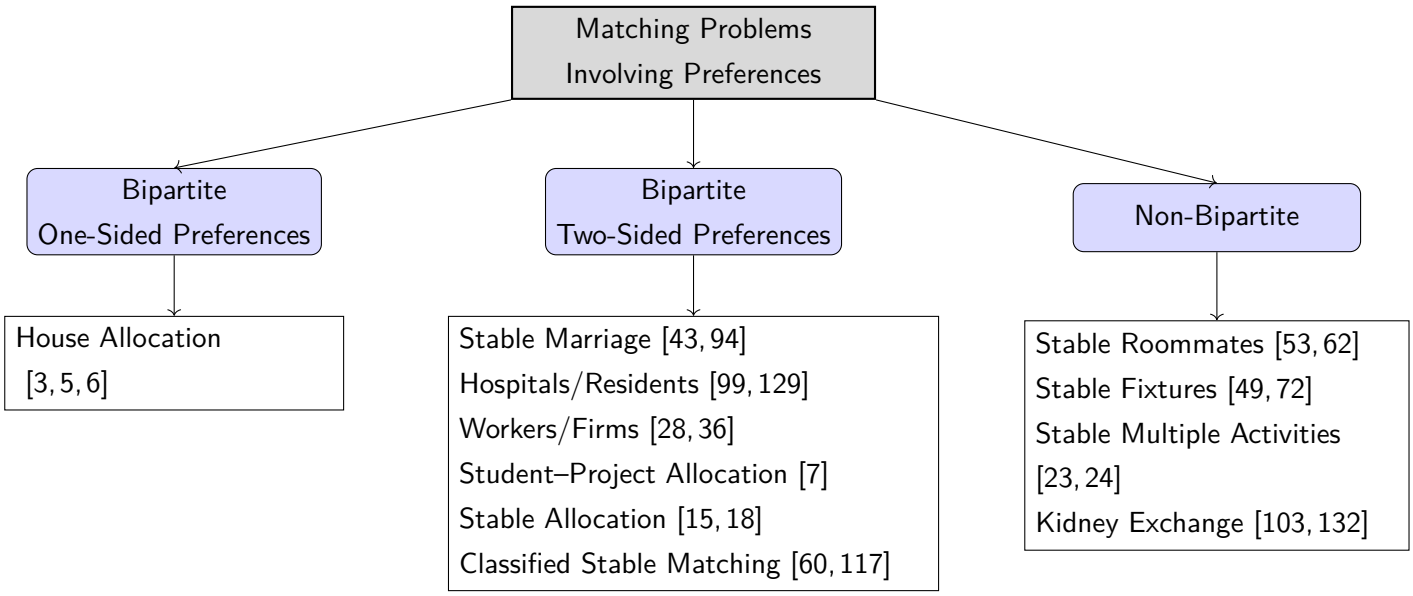

Figure 2.1: Classification of matching problems involving preferences.

2.1 The Stable Marriage Problem

The Stable Marriage problem (SM) involves two disjoint sets, typically referred to as men and women, where each individual has preferences over all members of the opposite set. Each man lists the women in order of preference, and each woman does the same for the men. The goal is

to find a matching M where every man is paired with exactly one woman, and every woman with exactly one man, such that there are no *blocking pairs*. A blocking pair is a man and woman who are not assigned to each other in M, but who both prefer each other to their current partners. This problem was first introduced by Gale and Shapley [43], and has been widely studied in areas like mathematics, economics, game theory, computer science, and physics [10, 17, 38, 84].

In the classical SM problem, each participant provides a strict ordering of *all* members of the opposite set, resulting in complete and strictly ordered preference lists. Several extensions of this model have been introduced to capture more general and realistic settings. In the *Stable Marriage problem with Incomplete lists* (SMI), participants find some members of the opposite set unacceptable, and therefore submit a preference list over only a subset of potential partners. The sets of men and women may also differ in size, so not all individuals are necessarily assigned. In the *Stable Marriage problem with Ties* (SMT), each person ranks all members of the opposite set, but is allowed to express indifference between some participants, resulting in *ties* in their preference lists. A more general extension, the *Stable Marriage problem with Ties and Incomplete lists* (SMTI), allows both ties and incomplete preference lists. These models are discussed in more detail in Section 2.1.2.

2.1.1 Formal definition

Formally, an instance of the Stable Marriage problem (SM) consists of two disjoint sets: a set $U = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women. Each man has a preference list ranking all women in W in order of preference, and each woman has a preference list ranking all men in U. These preference lists are complete and strictly ordered, meaning that each person ranks all members of the opposite set in a strict sequence with no ties. A man m is said to prefer woman w_x to woman w_y if w_x appears before w_y on his preference list; similarly, a woman w prefers man m_x to man m_y if m_x appears before m_y on her list.

A matching M is a set of n disjoint pairs, where each man is assigned to exactly one woman and each woman is assigned to exactly one man. In a given matching M, we write $M(m)$ to denote the woman assigned to man m, and $M(w)$ to denote the man assigned to woman w. A pair (m, w) that does not belong to the matching M is called a *blocking pair* if both m prefers w to $M(m)$, and w prefers m to $M(w)$. In this case, both individuals would strictly benefit from being matched with each other rather than with their current partners. A matching is said to be *stable* if it admits no blocking pair. If a blocking pair exists, then the matching is unstable.

Consider the example instance I_1 of the stable marriage problem shown in Figure 2.2, which consists of three men and three women. There are $3! = 6$ possible complete matchings, although not all of them are stable. It can be verified that the matchings $M = \{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$ and $M' = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$ are stable: in each case, there is no pair of agents who would prefer to be matched to each other over their assigned partners. In contrast, the matching

$M^* = \{(m_1, w_1), (m_2, w_3), (m_3, w_2)\}$ is unstable. This is because m_3 prefers w_1 to his assigned partner w_2, and w_1 also prefers m_3 to her assigned partner m_1, forming the blocking pair (m_3, w_1) in M^*.

Men's preferences				Women's preferences			
m_1:	w_1	w_2	w_3	w_1:	m_2	m_3	m_1
m_2:	w_2	w_3	w_1	w_2:	m_3	m_1	m_2
m_3:	w_3	w_1	w_2	w_3:	m_1	m_2	m_3

Figure 2.2: An instance I_1 of the Stable Marriage problem with 3 men and 3 women

2.1.1.1 The Gale-Shapley algorithm

It is well known that every instance of the SM problem admits at least one stable matching, and that such a matching can be found in time $O(n^2)$, where n in the number of participants involved, using the classical Gale–Shapley algorithm [43]. The Gale–Shapley algorithm begins with all men unassigned. At each step, every unassigned man proposes to the most-preferred woman on his list to whom he has not yet proposed. Each woman who receives one or more proposals compares them with her current partner (if any), tentatively accepts the most-preferred among these, and rejects the rest. Any man who is rejected becomes unassigned and, in a subsequent step, proposes to the next woman on his preference list to whom he has not yet proposed. The algorithm proceeds in this manner until every individual is assigned. Since each man proposes to women in order of his preference list and never proposes to the same woman more than once, the process is guaranteed to terminate after each man has either been accepted or has proposed to every woman on his list.

The Gale–Shapley algorithm contains an element of non-determinism, given that the order in which men make their proposals is not specified. However, Gale and Shapley [43] showed that, regardless of the order in which proposals are made, the algorithm always produces the same matching. Moreover, this version of the algorithm, commonly referred to as the *man-oriented* Gale–Shapley algorithm, yields the *man-optimal* stable matching M_0, in the sense that each man is assigned to the best partner he can obtain in any stable matching admitted by the instance. If the roles of men and women are reversed, so that women propose instead, the resulting *woman-oriented* Gale–Shapley algorithm produces the *woman-optimal* stable matching M_z, in which each woman is assigned her best possible partner among all stable matchings.

2.1.1.2 Extended Gale Shapley algorithm

An extended version of the Gale–Shapley algorithm (EGS) was introduced in [54] to compute a stable matching while also deleting certain pairs that cannot appear in any stable matching. As in the classical Gale–Shapley algorithm, each unassigned man proposes to the most-preferred

woman remaining on his list. If the woman is already assigned to another man m', she compares the two and retains the more preferred partner; the other becomes unassigned.

When a woman w becomes assigned to a man m, all men who appear after m on her list (called her *successors*) are deleted. In other words, for each such man m', m' is removed from w's preference list and w is removed from m''s preference list. If w was previously assigned to any of these men, those assignments are broken. It was shown that none of these successors can be matched with w in any stable matching. In the EGS algorithm, every proposal that occurs is accepted at the time it is made. Suppose a man m proposes to a woman w who is currently assigned to some man m'. Then w must prefer m to m'; otherwise, m would have been removed from her list when she was first assigned to m'. Since proposals are only made to acceptable partners who have not been deleted, each proposal is accepted. The algorithm terminates when all agents are assigned.

The resulting preference lists after all deletions have been made are called the *man-oriented Gale–Shapley lists* (MGS-lists). Running the algorithm with women proposing instead yields the *woman-oriented Gale–Shapley lists* (WGS-lists). The intersection of the MGS-list and WGS-list is the *Gale–Shapley list* (GS-list). This list contains all pairs that could possibly appear in a stable matching, although not all of them necessarily do. The pairs that actually occur in at least one stable matching of the instance are called *stable pairs*. In the man-optimal stable matching, each man is assigned to the first woman on his GS-list, and each woman is assigned to the last man on hers. The EGS algorithm runs in $O(n^2)$ time.

2.1.1.3 Multiple stable matchings

McVitie and Wilson [112, 113] observed that the man-optimal stable matching is also *woman-pessimal*, meaning each woman is assigned her worst partner among all stable matchings. Conversely, the woman-optimal stable matching is *man-pessimal*, with each man receiving his worst partner across all stable matchings in the instance. In addition to the man-optimal and woman-optimal stable matchings, a single instance of the SM problem may admit several others. For example, the instance I_1 shown in Figure 2.2 admits a third stable matching, $\{(m_1, w_2), (m_2, w_3), (m_3, w_1)\}$, in addition to the matchings M and M' highlighted earlier.

More generally, Knuth [85] observed that the number of stable matchings in an SM instance can grow exponentially with the size of the input n. This result was later strengthened by Irving and Leather [65], who showed that for each $n \geq 0$ that is a power of two, there exists an instance of size n with at least 2^{n-1} stable matchings. This means that any brute-force approach that attempts to examine all stable matchings in order to identify one that satisfies a given optimality criterion is impractical in the worst case. These observations motivated further investigation into the structure of the entire set of stable matchings, as discussed in Section 2.1.3.

Interestingly, McVitie and Wilson [112, 113] also presented a recursive version of the Gale–Shapley algorithm and proposed a method for enumerating all stable matchings in a given in-

stance of the stable marriage problem. Their approach repeatedly applies the Gale–Shapley algorithm and, for each stable matching found, generates new sub-instances by excluding certain pairs that appear in the previously computed matching. In this way, each stable matching is generated exactly once, and no matching is repeated. A different approach, based on backtracking search, was later developed by Wirth [147]. While Wirth's method is conceptually simpler, it is less efficient than the algorithm presented by McVitie and Wilson.

2.1.2 Extensions of the Stable Marriage problem (SM)

In this section, we consider three natural relaxations of the Stable Marriage problem. First, the number of men and women may be unequal, meaning that some agents will necessarily be unassigned. Second, agents may provide preferences over only a subset of the opposite set acceptable, resulting in *incomplete preference lists*. Third, agents may be indifferent between two or more potential partners, leading to *ties* in their preference lists.

These relaxations lead to the following extensions of the Stable Marriage problem. In the *Stable Marriage problem with Incomplete lists* (SMI), agents are allowed to declare some agents in the opposite set as unacceptable, and can be assigned only to those they consider acceptable. As a result, some agents may remain unassigned. In the *Stable Marriage problem with Ties* (SMT), each agent ranks all agents in the opposite set and may express indifference between some of them by including ties in their preference list. The most general variant is the *Stable Marriage problem with Ties and Incomplete lists* (SMTI), which permits both incomplete lists and ties in preferences.

2.1.2.1 Stable Marriage with Incomplete lists (SMI)

In the *Stable Marriage problem with Incomplete lists* (SMI), the sets of men and women, denoted U and W, need not be of equal size. Each man $m \in U$ provides a strictly ordered preference list over a subset of women in W, and each woman $w \in W$ similarly ranks a subset of men in U. Hence, agents may declare some members of the opposite set as *unacceptable*, indicating that they would rather remain unassigned than be matched with those individuals. A woman w is said to be *acceptable* to a man m if w appears on m's preference list, and vice versa. If m and w find each other acceptable, then (m, w) is called an *acceptable pair*.

A matching is a set of disjoint pairs (m, w), where each man $m \in U$ is assigned to at most one woman $w \in W$, and vice versa. Each pair in the matching must involve a man and a woman who are acceptable to one another. It is straightforward to see that in an smi instance, not all agents can be assigned, since the two sets may be of unequal size. Consequently, the notion of a blocking pair in SMI was redefined by Gusfield and Irving [54]. A pair (m, w) is a *blocking pair* with respect to a matching M if (m, w) is an acceptable pair and both (a) and (b) holds as follows:

(a) either m is unassigned in M, or prefers w to their assigned partner $M(m)$; and

(b) either w is unassigned in M, or prefers m to their assigned partner $M(w)$.

Again, a matching is *stable* if it admits no blocking pair. A stable matching is guaranteed to exist in any SMI instance, and can be found by a straightforward extension of the Gale–Shapley algorithm [54]. Moreover, the classical results concerning the man-optimal and woman-optimal stable matchings carry over naturally to this setting.

Although an arbitrary SMI instance may admit many stable matchings, Gale and Sotomayor [44] observed that all such matchings assign exactly the same subset of agents. Thus, if an agent is assigned (or unassigned) in one stable matching, then they are assigned (or unassigned) in all of them. We can think of the agents as being partitioned into two groups: those who are assigned in every stable matching and those who are never assigned. Consequently, to analyse the structure of stable matchings in SMI, it suffices to focus on the subset of agents who are assigned in every stable matching. Removing the unassigned agents, along with their entries in the preference lists of the assigned agents, does not affect the set of stable matchings admitted by the instance [54].

2.1.2.2 Stable Marriage with Ties (SMT)

In the *Stable Marriage problem with Ties* (SMT), agents are allowed to express indifference between some potential partners. This is done by introducing *ties* in the preference lists, where a tie represents a group of agents that an agent considers equally acceptable. We say that an agent *strictly prefers* one agent to another if the first appears earlier in the list and the two are not in the same tie. An agent is said to be *indifferent* between two other agents if those two agents appear together in the same tie.

When ties are allowed in preference lists, the standard definition of a blocking pair, where both agents strictly prefer each other to their current partners, no longer applies directly. This is because an agent may be indifferent between their assigned partner and another acceptable person. In such cases, it is not straightforward to determine whether reassigning the agent would actually improve their outcome. This leads to three notions of stability:

- **Weak Stability:** A matching M is *weakly stable* if there is no pair (m, w) such that both m and w strictly prefer each other to their partners in M.
- **Strong Stability:** A matching M is *strongly stable* if there is no pair $(m, w) \notin M$ such that either m strictly prefers w to $M(m)$, and w prefers or is indifferent between m and $M(w)$, or w strictly prefers m to $M(w)$, and m prefers or is indifferent between w and $M(m)$.
- **Super Stability:** A matching M is *super-stable* if there is no pair (m, w) such that both agents either strictly prefer each other to their assigned partners in M, or are indifferent between them and their current partners.

It follows that every super-stable matching is also strongly stable, and every strongly stable matching is also weakly stable. By arbitrarily breaking the ties in an instance I of SMT, one obtains an instance I' of SM, for which any stable matching is also a weakly stable matching in I. Consequently, a weakly stable matching in I can be found in $O(n^2)$ time using the Gale–Shapley algorithm, for example. However, it has been shown that an instance of SMT may admit neither a strongly stable matching nor a super-stable matching. Nonetheless, Irving [63] presented an $O(n^4)$ algorithm for determining whether a strongly stable matching exists and constructing one if it does, as well as an $O(n^2)$ algorithm for deciding whether a super-stable matching exists and returning such a matching when it does.

2.1.2.3 Stable Marriage with Ties and Incomplete Lists (SMTI)

The *Stable Marriage problem with Ties and Incomplete lists* (SMTI) generalises both SMI and SMT. In this setting, agents may omit some members of the opposite set from their preference lists, as in SMI, and may express indifference between two or more acceptable partners by placing them in a tie, as in SMT. The three stability notions defined for SMT, namely weak, strong, and super stability, extend naturally to SMTI.

To illustrate these blocking pair notions more concretely, consider instance I_2 shown in Figure 2.3, which involves two men and two women, with a single tie appearing in the preference list of m_2 (In figures involving SMTI instances, ties are indicated using brackets). The matching $M = \{(m_1, w_1), (m_2, w_2)\}$ is weakly stable, since no pair of agents prefers each other to their partners in M. However, M is not strongly stable. The pair (m_2, w_1) blocks M because m_2 is indifferent between w_1 and w_2, while w_1 prefers m_2 to her partner in M, namely m_1. In contrast, the matching $M' = \{(m_2, w_1)\}$ is weakly stable, strongly stable, and super-stable.

Men's preferences	Women's preferences
m_1: w_1	w_1: m_2 m_1
m_2: $(w_1 w_2)$	w_2: m_2

Figure 2.3: Instance I_2 of SMTI with two men and two women.

Similar to SMI, we note that a weakly stable matching in a given SMTI instance always exists. This can be obtained by arbitrarily breaking ties and then applying the Gale–Shapley algorithm to the resulting SMI instance. However, Manlove *et al*. [100] showed that the manner in which ties are resolved can lead to weakly stable matchings of different sizes. For example, in Figure 2.3, instance I_2 admits two weakly stable matchings, M and M', of sizes 2 and 1, respectively. They further proved that MAX-SMTI, the problem of computing a maximum-size weakly stable matching, is NP-hard, even in cases where ties appear only at the ends of preference lists, occur on one side only, and each list contains at most one tie of length two.

Subsequently, Irving *et al*. [68] showed that MAX-SMTI is solvable in polynomial time when each man's preference list contains at most two women, even if the women's lists are of unbounded length. However, they also showed that the problem remains NP-hard when each man's preference list has length at most 3, and remains so even if each woman's list also has length at most 3. Moreover, the problem is not approximable within any factor $\delta > 1$, even when each woman's list is of length at most 4. Furthermore, Yanagisawa [148] proved that MAX-SMTI is not approximable within a factor of $\frac{33}{29}$ unless P = NP.

Panda and Sachin [127] strengthened these results by examining the structure of preference lists. They showed that the decision variant of MAX-SMTI, denoted COM-SMTI, which asks whether there exists a complete weakly stable matching in a given SMTI instance, [2] remains NP-complete even when each preference list consists of consecutive members with respect to some fixed ordering of the set of men and the set of women. They also identified a restricted case, denoted SMTI-STEP, in which there exist orderings of the men and women such that each man m_i finds acceptable exactly those women w_j with $j \leq i$. Under this condition, they showed that the COM SMTI problem can be solved in $O(n^2)$ time.

Given the negative complexity results, several heuristic strategies for MAX-SMTI have been presented, many of which are based on local search techniques [47,57,116]. Alongside these heuristics, approximation algorithms have also been developed [55, 67, 81, 110, 126]. The best known approximation algorithm for the case where ties are allowed on both sides has an approximation factor of $\frac{3}{2}$ [110]. For the case where ties occur only on one side, Iwama *et al*. [75] gave a $\frac{25}{17}$-approximation algorithm, which was improved to $\frac{22}{15}$ by Huang and Kavitha [61]. Subsequently, Radnai [128] presented a $\frac{41}{28}$-approximation algorithm, which was later improved to $\frac{19}{13}$ by Dean *et al.* [32]. The current best approximation factor, $1 + \frac{1}{e}$, was obtained by Lam and Plaxton [91].

Matsuyama and Miyazaki [108] noted that it is generally difficult to evaluate the quality of approximation algorithms experimentally, since computing an optimal solution for large instances of MAX-SMTI is often infeasible. To address this, they considered the problem of generating MAX-SMTI instances with known optimal solutions and explored whether an instance generation algorithm could be designed to produce such examples. They showed that if an instance generator could produce all such instances in polynomial-time, then NP= coNP [3], which is widely believed to be unlikely. Moreover, they proposed three instance generators that construct restricted versions of SMTI instances, for which the optimal solution is known.

Similar to the SMT setting, an instance of SMTI may admit no strongly stable or super-stable matching [54]. Nonetheless, Manlove [97] showed that, for each of these stronger notions of stability, the set of agents can be partitioned into those who are assigned in every such matching

[2] A *complete weakly stable matching* is a weakly stable matching in which every man and every woman is matched.

[3] coNP is the class of decision problems where, for every input for which the answer is NO, there exists a certificate that can be verified in polynomial time

and those who are unassigned in all. He also developed polynomial-time algorithms to determine whether a strongly stable or super-stable matching exists in a given instance, and to construct one if it does. These algorithms run in $O(n^4)$ and $O(n^2)$ time, respectively.

2.1.3 Structure of the set of stable matchings in SM and its extensions

In Section 2.1, we noted that an instance of SM admits both a man-optimal and a woman-optimal stable matching, denoted M_0 and M_z, respectively, depending on whether the man-oriented or woman-oriented version of the Gale–Shapley algorithm is applied. When $M_0 = M_z$, the instance admits a unique stable matching, as this is the only case in which every man's best partner is also his worst. If $M_0 \neq M_z$, then the instance may admit additional stable matchings besides M_0 and M_z. These two matchings correspond to the extremal elements in the set of stable matchings, with M_0 being optimal for all men and M_z optimal for all women. We note that the set of all stable matchings in SM, denoted $\mathcal{M}$, exhibits a rich underlying structure. In this section, we explore compact representations of $\mathcal{M}$ and discuss the algorithmic implications of this structure.

2.1.3.1 Lattice structure in SM

Let I be an instance of SM, and let $\mathcal{M}$ denote the set of all stable matchings in I. We define a partial order on $\mathcal{M}$ as follows. Let M and M' be two stable matchings in $\mathcal{M}$. We say that M *dominates* M', denoted $M \succeq M'$, if for every man m, either $M(m) = M'(m)$ (that is, m is assigned the same partner in both matchings), or m prefers $M(m)$ to $M'(m)$. Intuitively, M *dominates* M' if every man prefers his partner in M at least as much as his partner in M'. Under this dominance relation, the structure $(\mathcal{M}, \succeq)$ forms a partial order.

Knuth [85], attributing the observation to John Conway, noted that if each man is assigned the more (or less) preferred of his two partners in two stable matchings M and M', the resulting assignment is itself a stable matching. Consequently, the set $\mathcal{M}$, ordered by $\succeq$, forms a *distributive lattice* (see Definition 4.2.3), where the *meet* (respectively, *join*) of two stable matchings yields another stable matching in which each man is assigned the more (respectively, less) preferred of his two partners. Moreover, the maximum and minimum elements of this lattice correspond to the man-optimal and woman-optimal stable matchings, respectively. A formal proof of this result can be found in [54, Section 1.3.1]. For completeness, we restate the key results below.

Lemma 2.1.1 ([54]). *Let M and M' be two stable matchings in a given instance of* SM. *Define their meet, $M \wedge M'$, as the matching obtained by assigning each man the better of his partners in M and M'. Then $M \wedge M'$ is a stable matching.*

Lemma 2.1.2 ([54]). *Let M and M' be two stable matchings in a given instance of* SM. *Define their join, $M \vee M'$, as the matching obtained by assigning each man the worse of his partners in M and M'. Then $M \vee M'$ is a stable matching.*

Lemmas 2.1.1 and 2.1.2 imply the following result:

Theorem 2.1.1 ([54]). *Let $\mathcal{M}$ be the set of all stable matchings in a given instance of the stable marriage problem. Define a partial order $\succeq$ on $\mathcal{M}$ such that $M \succeq M'$ if and only if each man prefers his partner in M to his partner in M', or is assigned the same partner in both matchings. Then, the poset $(\mathcal{M}, \succeq)$ forms a distributive lattice. In this lattice, the meet $M \wedge M'$ corresponds to the matching where each man gets his more preferred partner between M and M', while the join $M \vee M'$ corresponds to the matching where each man gets his less preferred partner between M and M'.*

We note that all results concerning the lattice structure extend to the SMI setting. The lattice structure of the set of stable matchings provides a useful foundation for the design of algorithms for related problems, such as enumerating all stable matchings of a given instance or identifying one that satisfies additional properties.

However, this structure alone does not immediately lead to efficient algorithms. Since the set $\mathcal{M}$ of stable matchings may be exponentially large, any algorithm that explicitly constructs the entire lattice will, in the worst case, require exponential time. To address this, Irving and Leather [65] introduced the *rotation poset*, a polynomial-sized structure that compactly encodes all stable matchings of an instance of SM. This poset essentially captures all the different ways in which one can *navigate* the lattice of stable matchings, allowing transitions from one matching to another without generating the full lattice.

2.1.3.2 Rotations in SM

For a stable matching M in an instance I of SM, let $s_M(m)$ denote the next woman w on m's preference list, that appears after $M(m)$, such that w prefers m to her current partner $M(w)$, if such a woman exists. A *rotation* ρ is an ordered sequence of man–woman pairs $\{(m_0, w_0), \ldots, (m_{r-1}, w_{r-1})\}$ such that, for each i $(0 \leq i \leq r-1)$, the pair $(m_i, w_i) \in M$, and $w_{i+1} = s_M(m_i)$, where all indices are taken modulo r. We say that ρ is *exposed* in M if every pair in ρ is included in M. If ρ is exposed in M, we may *eliminate* ρ. To eliminate a rotation is to reassign each man m_i to w_{i+1}, and all agents not involved in the rotation are unaffected. The resulting matching is denoted by M/ρ, and is guaranteed to be stable [54, 65].

Note that eliminating a rotation ρ causes the men involved in ρ to be strictly worse off, and the women to be strictly better off. Moreover, every stable matching except the woman-optimal matching M_z admits at least one exposed rotation [54,65]. Let M be a stable matching in a given SM instance I. If $M \neq M_z$, then there is at least one rotation exposed in M. Let $\{\rho_0, \rho_1, \ldots, \rho_k\}$ denote the set of rotations exposed in M. Eliminating one of these rotations, say ρ_0, yields a new stable matching M/ρ_0. In this new matching, the remaining rotations $\rho_1, \ldots, \rho_k$ remain exposed,

and additional rotations that were not exposed in M may now become exposed. By repeatedly eliminating exposed rotations in this way, we move from one stable matching to another. At each step, the set of exposed rotations will change.

To illustrate the concepts in Section 2.1.3.1 and 2.1.3.2, we present an example SM instance I_3 showing the lattice of stable matchings and the corresponding rotations in I_3.

Example: The Stable Marriage instance I_3, shown in Figure 2.4, consists of 8 men and 8 women. This instance admits 8 stable matchings, presented in Table 2.1, with M_1 as the man-optimal matching and M_8 as the woman-optimal matching. The lattice structure for the set of all stable matchings in I_3 is shown in Figure 2.5, with M_1 at the top and M_8 at the bottom of the lattice. The lattice structure forms a directed graph where each vertex represents a stable matching. There is a directed edge from a vertex M to a vertex M' (with $M \neq M'$) if $M \preceq M'$ and no intermediate matching $\hat{M}$ (distinct from both M and M') satisfies $M \preceq \hat{M} \preceq M'$. For example, using the definitions of meet and join from Lemmas 2.1.1 and 2.1.2, it can be verified that $M_2 = M_3 \wedge M_4$ and $M_5 = M_3 \vee M_4$. Moreover, M_2 dominates both M_3 and M_4, as well as several other stable matchings.

Men's preferences	Women's preferences
m_1: w_5 w_7 w_1 w_2 w_6 w_8 w_4 w_3	w_1: m_5 m_3 m_7 m_6 m_1 m_2 m_8 m_4
m_2: w_2 w_3 w_7 w_5 w_4 w_1 w_8 w_6	w_2: m_8 m_6 m_3 m_5 m_7 m_2 m_1 m_4
m_3: w_8 w_5 w_1 w_4 w_6 w_2 w_3 w_7	w_3: m_1 m_5 m_6 m_2 m_4 m_8 m_7 m_3
m_4: w_3 w_2 w_7 w_4 w_1 w_6 w_8 w_5	w_4: m_8 m_7 m_3 m_2 m_4 m_1 m_5 m_6
m_5: w_7 w_2 w_5 w_1 w_3 w_6 w_8 w_4	w_5: m_6 m_4 m_7 m_3 m_8 m_1 m_2 m_5
m_6: w_1 w_6 w_7 w_5 w_8 w_4 w_2 w_3	w_6: m_2 m_8 m_5 m_3 m_4 m_6 m_7 m_1
m_7: w_2 w_5 w_7 w_6 w_3 w_4 w_8 w_1	w_7: m_7 m_5 m_2 m_1 m_8 m_6 m_4 m_3
m_8: w_3 w_8 w_4 w_5 w_7 w_2 w_6 w_1	w_8: m_7 m_4 m_1 m_5 m_2 m_3 m_6 m_8

Figure 2.4: An instance I_3 of SM, adapted from Gusfield and Irving [54, page 69]

Matching	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
M_1	w_5	w_3	w_8	w_6	w_7	w_1	w_2	w_4
M_2	w_8	w_3	w_5	w_6	w_7	w_1	w_2	w_4
M_3	w_3	w_6	w_5	w_8	w_7	w_1	w_2	w_4
M_4	w_8	w_3	w_1	w_6	w_7	w_5	w_2	w_4
M_5	w_3	w_6	w_1	w_8	w_7	w_5	w_2	w_4
M_6	w_8	w_3	w_1	w_6	w_2	w_5	w_7	w_4
M_7	w_3	w_6	w_1	w_8	w_2	w_5	w_7	w_4
M_8	w_3	w_6	w_2	w_8	w_1	w_5	w_7	w_4

Table 2.1: The eight stable matchings admitted by instance I_3, where each entry shows the woman assigned to each man.

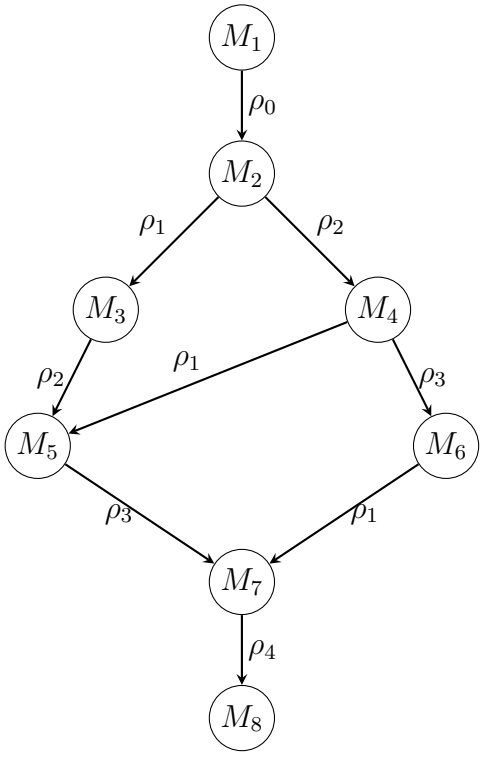

Figure 2.5: Lattice of stable matchings and corresponding rotations in instance I_3.

We note that instance I_3 admits a total of five rotations, given below:

$$\rho_0 = \{(m_1, w_5), (m_3, w_8)\},$$
$$\rho_1 = \{(m_1, w_8), (m_2, w_3), (m_4, w_6)\},$$
$$\rho_2 = \{(m_3, w_5), (m_6, w_1)\},$$
$$\rho_3 = \{(m_7, w_2), (m_5, w_7)\},$$
$$\rho_4 = \{(m_3, w_1), (m_5, w_2)\}.$$

We now illustrate how eliminating a rotation leads to a new stable matching. Starting from the man-optimal stable matching for instance I_3,

$$M_1 = \{(m_1, w_5), (m_2, w_3), (m_3, w_8), (m_4, w_6), (m_5, w_7), (m_6, w_1), (m_7, w_2), (m_8, w_4)\},$$

the only rotation exposed in M_1, as shown in Figure 2.5, is $\rho_0 = \{(m_1, w_5), (m_3, w_8)\}$. Eliminating ρ_0 involves moving m_1 to woman w_8 and m_3 to woman w_5. As a result, we obtain the new stable matching

$$M_2 = \{(m_1, w_8), (m_2, w_3), (m_3, w_5), (m_4, w_6), (m_5, w_7), (m_6, w_1), (m_7, w_2), (m_8, w_4)\}.$$

In M_2, the rotations ρ_1 and ρ_2 are now exposed. Figure 2.5 shows the rotations exposed in each matching, with the corresponding rotations labelled on the edges. By successively eliminating an exposed rotation at each step, we obtain the next stable matching in the lattice.

2.1.3.3 Rotation poset

Let R denote the set of all rotations that are exposed in at least one stable matching of a given instance I. Irving and Leather [65] showed that any two rotations in R are either identical or disjoint; that is, no man–woman pair appears in more than one rotation. Furthermore, there is a natural partial order on R. Specifically, if a rotation ρ must be eliminated before another rotation ρ' can be exposed, then we say that ρ *precedes* ρ', denoted $\rho \prec \rho'$. This partial order defines the *rotation poset* $\Pi = (R, \prec)$. It is known that the number of rotations in Π is bounded by $O(m)$, where m is the total length of all preference lists in the instance.

A subset $R' \subseteq R$ is said to be closed if, for every rotation $\rho \in R'$, all rotations ρ' with $\rho' \prec \rho$ are also contained in R'. Irving and Leather [65] (see also [54, Section 2.5.4]) showed that there is a one-to-one correspondence between the set of stable matchings and the set of closed subsets of Π. Starting from any stable matching M in I, we can eliminate the rotations in any closed subset R', in any order that respects the partial order, and obtain a different stable matching. In fact, every stable matching of I can be obtained by eliminating a closed subset of rotations starting from the man-optimal stable matching. Thus, the set Π encodes the entire set $\mathcal{M}$ of stable matchings.

Gusfield and Irving [54] showed that by constructing the *rotation digraph* $G(\mathcal{M})$, which is derived from the rotation poset, it is possible to enumerate all stable matchings in time $O(m + n|\mathcal{M}|)$, where n is the number of agents, m is the total length of all preference lists, and $|\mathcal{M}|$ is the number of stable matchings in the instance. Moreover, each edge in the lattice of stable matchings corresponds exactly to the elimination of a single rotation in the rotation digraph. The rotation poset and its associated digraph have also been used to derive complexity bounds and design efficient algorithms for several stable marriage variants, including the egalitarian and minimum

regret stable matching problems [52].

2.1.3.4 Optimal stable matchings

In certain applications, it is desirable to impose additional optimality criteria on stable matchings in order to improve overall satisfaction or fairness. One such criterion is the *egalitarian* stable matching, which aims to minimise the total dissatisfaction across all participants. For each pair (m_i, w_j) in a stable matching M, we define the rank of m_i in M to be the position of w_j on m_i's preference list, and the rank of w_j in M to be the position of m_i on w_j's preference list; the sum of all such ranks over all assigned pairs gives the total weight of M. An egalitarian stable matching is a stable matching with minimum possible weight. In the SM setting, an efficient algorithm for finding such a matching, which leverages the distributive lattice structure of stable matchings, was described by Irving *et al.* [64, 66]. Another optimality notion is the *minimum regret* stable matching, which focuses on minimising the dissatisfaction of the worst-off participant. A polynomial-time algorithm for finding a minimum regret stable matching was proposed in [52].

2.1.3.5 Polyhedral characterization of stable marriages

The structure of the set of stable matchings has been investigated using a polyhedral approach. In particular, the set of stable marriages can be described as the set of extreme points of a polytope known as the stable marriage polytope. This polytope is defined by a set of linear inequalities that capture the matching constraints (ensuring that each participant is matched to at most one partner) and the stability constraints (preventing blocking pairs). Vande Vate [144] was the first to describe this polytope explicitly and to show that each stable matching corresponds to an extreme point of the polytope. Later, Rothblum [135] provided a simplified proof of this result. Building on these results, Gusfield and Irving [54] introduced an alternative characterization using rotations, which also proves that the set of stable matchings corresponds exactly to the extreme points of the stable marriage polytope.

This polyhedral approach makes it possible to derive results for the SM problem using linear programming techniques. Furthermore, the formulation of the stable matching polytope enables the construction of polynomial-time reductions between the stable marriage problem and other combinatorial optimisation problems, which in turn helps identify and analyse problems that are structurally equivalent to the stable marriage problem. Two problems A and B are said to be *structurally equivalent* if there exists a structure-preserving reduction between them, whereby a solution is feasible for an instance of A if and only if its corresponding solution is feasible for the corresponding instance of B, and other essential structural properties are preserved; for more details, see [13]. In the context of the stable marriage, Gusfield and Irving [54] showed, using the rotation digraph, that the stable marriage problem is structurally equivalent to the minimum s-t cut problem.

2.1.3.6 Structure of strongly and super-stable matchings

The sets of strongly-stable and super-stable matchings in SMTI also possess well-defined structural properties. Manlove [98] showed that the set of strongly stable matchings in SMTI forms a finite distributive lattice. Later, Kunysz *et al.* [87] presented two characterisations of the set of strongly stable matchings in SMTI. The first is based on the notion of irreducible matchings. For each stable pair (a, b), meaning an acceptable pair that can appear in some strongly stable matching, they define a unique strongly stable matching that is best for all men among those matchings containing (a, b). This matching is called the irreducible matching corresponding to (a, b). All such irreducible matchings can be computed in $O(nm^2)$ time, where n and m denote the numbers of vertices and edges in the graph G representing the underlying SMTI instance. The second characterisation uses rotations, which generalise the rotation concept from SM to instances with ties and incomplete lists. They also describe how to compute the rotation poset in $O(nm)$ time.

Similarly, Speiker [138] demonstrated that the set of super-stable matchings in SMTI also forms a distributive lattice, with an alternative proof provided by Manlove [98]. Scott [137] introduced the concept of meta-rotations for super-stable matchings, which can be constructed in $O(m^2)$ time, and established a one-to-one correspondence between super-stable matchings and the closed subsets of the associated poset. More recently, Hu and Garg [59] presented a simpler characterisation of the set of super-stable matchings, based on rotations, which can be constructed in $O(mn)$ time. In addition, Kunysz [86] provided a polyhedral characterization of the set of all strongly stable matchings and proved that the strongly stable matching polytope is integral. On the other hand, Hu and Garg [59] presented a polyhedral characterization for the set of all super-stable matchings and showed that the super-stable matching polytope is integral, using Hall's theorem.

2.2 The Hospitals/Residents problem (HR)

The Hospital/Residents problem (also known as the College Admissions problem) is a many-to-one generalisation of the Stable Marriage problem with Incomplete lists (SMI), first introduced by Gale and Shapley [43]. The agents involved are residents and hospitals, with each hospital having a fixed number of available positions (capacity). A matching is an assignment of residents to hospitals such that each hospital does not exceed its capacity and each resident is assigned to at most one hospital. Similar to SMI, the goal is to find a stable matching.

2.2.1 Formal definition

Formally, an instance of HR consists of a set R of residents and a set H of hospitals. Each resident $r_i \in R$ ranks a subset of hospitals they find acceptable in strict preference order; this defines their preference list. If a hospital h_j appears on r_i's preference list, we say that r_i finds h_j acceptable.

Given two hospitals $h_j, h_k \in H$, if h_j precedes h_k on r_i's preference list, we say that r_i prefers h_j to h_k. Each hospital $h_j \in H$ also ranks a subset of residents it finds acceptable, which forms its preference list. The same notion of preference applies to hospitals: given two residents $r_i, r_l \in R$, if r_i precedes r_l on h_j's preference list, then h_j prefers r_i to r_l. A resident-hospital pair (r_i, h_j) is called an *acceptable pair* if r_i finds h_j acceptable and h_j finds r_i acceptable. Each hospital h_j has a capacity $c_j \in \mathbb{Z}^+$, representing the maximum number of residents it can accommodate.

A matching M is an assignment of acceptable resident–hospital pairs such that each resident is assigned to at most one hospital, and no hospital is assigned more residents than its capacity. If $(r_i, h_j) \in M$, we say that h_j is assigned r_i in M, and we denote by $M(h_j)$ the set of residents assigned to h_j. Similarly, $M(r_i)$ denotes the hospital assigned to r_i. We write $|M(h_j)|$ to denote the number of residents assigned to h_j. A hospital h_j is said to be *undersubscribed* if $|M(h_j)| < c_j$, *full* if $|M(h_j)| = c_j$, and *oversubscribed* if $|M(h_j)| > c_j$.

A pair (r_i, h_j) not in M is called a *blocking pair* with respect to M if it is an acceptable pair and both of the following conditions hold:

(a) either r_i is unassigned in M or prefers h_j to $M(r_i)$;

(b) either h_j is undersubscribed, or is full in M and prefers r_i to its worst-ranked resident in $M(h_j)$.

A matching M is said to be *stable* if it admits no blocking pair.

Similar to SMI, Gale and Shapley [43] showed that a stable matching always exists in any instance of HR, and it can be found in polynomial time using either the resident-oriented or the hospital-oriented Gale–Shapley algorithm. Moreover, the algorithm produces a stable matching that is resident-optimal (and hospital-pessimal) or hospital-optimal (and resident-pessimal), and there may be multiple stable matchings in a given instance of HR.

2.2.2 Extensions of the Hospitals/Residents problem (HR)

A natural generalisation of HR allows residents (respectively, hospitals) to express indifference between two or more hospitals (respectively, residents) in their preference lists, in form of ties. This extension is known as the Hospitals/Residents problem with Ties (HRT). The HRT model extends SMTI, and the three stability definitions from SMTI, namely weak stability, strong stability, and super-stability, were generalised to HRT by Irving *et al.* [69,70]. A weakly stable matching in HRT is defined analogously to stability in HR. For formal definitions of strong and super stability in HRT, we refer the reader to Manlove [94, Section 1.3.5]. A polynomial-time algorithm for finding a strongly stable matching, or reporting that none exists, in an HRT instance was given in [69]. Likewise, an algorithm for finding a super stable matching was presented in [70]. Since HRT is an extension of SMTI, it follows that every instance of HRT admits a weakly stable matching [69],

and weakly stable matchings in HRT may differ in size. Moreover, finding a maximum weakly stable matching in an instance of HRT (MAX-HRT) is NP-hard [73].

Many of the approximability results and approximation algorithms for SMTI can be generalised to the HRT setting [94, 100]. Király [82] developed a $\frac{3}{2}$-approximation algorithm for MAX-HRT, and Yanagisawa [148] showed that there is no approximation algorithm for MAX-HRT with an approximation factor better than $\frac{33}{29}$, unless $\mathrm{P} = \mathrm{NP}$. Other approaches that have been explored to solve MAX-HRT include integer programming techniques [89,109], heuristic algorithms [21,120], and parameterised complexity [105].

The Hospitals/Residents problem with Couples (HRC) [101] extends HR by allowing certain residents to apply jointly as couples (for example, couples may wish to be assigned to hospitals that are close to each other). This is achieved by allowing each couple to submit a joint preference list over pairs of hospitals. The definition of stability in HR can be extended to HRC [94]. In this setting, a stable matching may not exist; however, it is possible to seek an *almost-stable* matching, that is, a matching minimising the number of blocking pairs. Manlove *et al.* [101] showed that finding such a matching is NP-hard. In a restricted variant of HRC where each single resident's preference list contains at most α hospitals, each couple's list contains at most β pairs of hospitals, and each hospital's list contains at most γ residents (referred to as (α, β, γ)-HRC), Manlove and McDermid [111] showed that deciding whether an instance of $(3, 2, 4)$-HRC admits a stable matching is NP-complete. Further results on restrictions, tractability, and hardness for HRC can be found in [29].

2.2.3 Structure of the set of stable matchings in HR

Similar to SM, an instance I of HR may admit many stable matchings in addition to the resident-optimal and hospital-optimal stable matchings. Roth and Sotomayor [133] established the following structural properties, known as the *Rural Hospitals Theorem* 2.2.1, regarding the set of stable matchings in I.

Theorem 2.2.1 (Rural Hospitals Theorem [44, 129, 130, 133])**.** *Let I be an instance of* HR*. Then, the following properties hold in I:*

- *The same set of residents are assigned in all stable matchings;*
- *Each hospital is assigned the same number of residents in all stable matchings;*
- *Any hospital that is undersubscribed in one stable matching is assigned exactly the same set of residents in every stable matching.*

Roth and Sotomayor [133] proved that any two stable matchings can be directly compared based on the hospitals' preferences. Suppose M and M' are two stable matchings for an instance of

HR, and hospital h is assigned different sets of residents in these matchings. If h prefers its worst resident in $M(h) \setminus M'(h)$ to its worst resident in $M'(h) \setminus M(h)$, then h prefers all residents in $M(h)$ to all residents in $M'(h) \setminus M(h)$. This result implies that, after excluding residents assigned to h in both M and M', the hospital prefers all residents in one matching to all residents in the other. It is therefore straightforward to define what it means for the hospitals as a group to prefer one matching over another. This observation, together with a similar resident-oriented relation, has been used to show that the set of stable matchings forms a finite distributive lattice [54].

Bansal [14] extended the notion of rotations to the many-to-many stable matching setting, in which agents on both sides can be matched to multiple partners. In this setting, each agent has a capacity specifying the maximum number of agents they can be assigned. By introducing *meta-rotations*, the authors showed how all stable matchings in a given instance can be enumerated, and developed a polynomial-time algorithm for finding a stable matching that is optimal in the sense of minimising the total dissatisfaction score[4]. Cheng *et al.* [26] specialised meta-rotations to the Hospitals/Residents problem (HR) and showed that the structural results developed for the many-to-many setting also hold in HR. They used meta-rotations to identify feasible stable matchings, that is, stable matchings that satisfy additional constraints with a so-called *identification property*. Furthermore, they introduced generalised notions of egalitarian and minimum regret stable matchings for HR, and developed polynomial-time algorithms for finding these matchings.

2.3 The Student-Project Allocation problem (SPA)

The Student-Project Allocation problem (SPA) is a generalisation of the Hospitals/Residents problem (HR) involving three sets of entities: students, projects, and lecturers. Each project is offered by exactly one lecturer, and both lecturers and projects have capacity constraints indicating the maximum number of students they can accommodate. A matching is an assignment of students to projects based on preferences, such that each student is assigned to exactly one project, and the capacity of both projects and lecturers are not exceeded. Applications of SPA in academic settings include matching schemes in the School of Computing at the University of Glasgow [88], the Department of Civil and Environmental Engineering at the University of Southampton [11, 56], and the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore [141].

In the SPA model, students have preferences over projects, while the presence and nature of lecturers' preferences give rise to three different variants of the model. In the first variant, known as *the Student-Project Allocation problem with lecturer preferences over Students* (SPA-S) [8], each lecturer provides preferences over the students who find at least one of their offered projects

[4]The total dissatisfaction score is defined as the sum of the ranks of assigned partners in each individual's preference list; a lower rank means higher preference.

acceptable. In the second variant, called the *Student-Project Allocation problem with lecturer preferences over Projects* (SPA-P) [102], each lecturer specifies a strict order of preference over the projects they offer. A third, more general variant allows lecturers to rank student-project pairs in strict order of preference, giving rise to the *Student-Project Allocation problem with lecturer preferences over student-project pairs* (SPA-(S,P)) [37]. In SPA-(S,P), each student-project pair on a lecturer's list involves a project offered by that lecturer and a student who finds that project acceptable. Further details on SPA-(S,P) can be found in [37, 94].

In this thesis, we focus on SPA-P and SPA-S, which we review in the following sections.

2.3.1 Student-Project Allocation with lecturer preferences over Students (SPA-S)

In this section, we formally define the SPA-S model introduced above, describe the notion of stability in this model, and present known results related to it.

2.3.1.1 Formal definition

Formally, an instance I of SPA-S consists of a set of students $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_1}\}$, a set of projects $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_2}\}$, and a set of lecturers $\mathcal{L} = \{l_1, l_2, \ldots, l_{n_3}\}$. Each project $p_j \in \mathcal{P}$ is offered by exactly one lecturer. For each lecturer $l_k \in \mathcal{L}$, let $\mathcal{P}_k \subseteq \mathcal{P}$ denote the set of projects offered by l_k. The sets $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{n_3}$ form a partition of $\mathcal{P}$; that is, each project is offered by exactly one lecturer. Moreover, each project p_j has a capacity $c_j \in \mathbb{Z}^+$, representing the maximum number of students that can be assigned to p_j. Similarly, each lecturer l_k has a capacity $d_k \in \mathbb{Z}^+$, indicating the maximum number of students they can supervise. Each student $s_i \in \mathcal{S}$ finds certain projects acceptable; this set of acceptable projects is denoted $\mathcal{A}_i \subseteq \mathcal{P}$. The student ranks the projects in $\mathcal{A}_i$ in strict order of preference, forming their preference list. Similarly, each lecturer $l_k \in \mathcal{L}$ provides a strict preference ordering over the students who find at least one project in $\mathcal{P}_k$ acceptable. We denote this preference list by $\mathcal{L}_k$.

A pair $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$, where project p_j is offered by lecturer l_k, is called an *acceptable pair* if and only if s_i finds p_j acceptable and l_k finds s_i acceptable. Formally, this means $p_j \in \mathcal{A}_i$ and $s_i \in \mathcal{L}_k$. For each lecturer l_k, we assume that $\max\{c_j : p_j \in \mathcal{P}_k\} \leq d_k \leq \sum_{p_j \in \mathcal{P}_k} c_j$, where $\mathcal{P}_k$ is the set of projects offered by lecturer l_k. This means that the capacity d_k of l_k is at least the largest capacity among the projects in $\mathcal{P}_k$ and at most the sum of the capacities of all projects in $\mathcal{P}_k$. We denote by $\mathcal{L}_j^k$ the projected preference list of lecturer l_k for project p_j. This list is obtained from $\mathcal{L}_k$ by removing all students who do not find p_j acceptable; the order of the remaining students is inherited from $\mathcal{L}_k$.

An assignment M for an instance I of SPA-S is a set of acceptable pairs $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ such that $(s_i, p_j) \in M$ only if $p_j \in \mathcal{A}_i$. If $(s_i, p_j) \in M$ and l_k is the lecturer offering p_j, we say that

s_i is *assigned to* project p_j and lecturer l_k; equivalently, p_j and l_k are assigned to s_i. The *size* of M, denoted $|M|$, is the number of student-project pairs it contains. We denote by $M(s_i)$ the project assigned to s_i (if any), by $M(p_j)$ the set of students assigned to p_j, and by $M(l_k)$ the set of students assigned to lecturer l_k. A project p_j is *undersubscribed*, *full*, or *oversubscribed* in M if $|M(p_j)|$ is less than, equal to, or greater than its capacity c_j, respectively. Similarly, a lecturer l_k is *undersubscribed*, *full*, or *oversubscribed* depending on whether $|M(l_k)|$ is less than, equal to, or greater than its capacity d_k, respectively. Furthermore, a project p_j is *non-empty* if $|M(p_j)| > 0$.

Finally, an assignment M is a *matching* if:

- each student is assigned to at most one project, i.e., $|M(s_i)| \leq 1$ for each $s_i \in \mathcal{S}$;
- no project exceeds its capacity, i.e., $|M(p_j)| \leq c_j$ for each $p_j \in \mathcal{P}$;
- no lecturer exceeds their capacity, i.e., $|M(l_k)| \leq d_k$ for each $l_k \in \mathcal{L}$.

Definition 2.3.1 (Stability in SPA-S [7])**.** Let I be an instance of SPA-S and M a matching in I. An acceptable pair $(s_i, p_j) \notin M$ is a *blocking pair* if:

(S1) s_i is either unassigned in M, or prefers p_j to their assigned project $M(s_i)$,

and one of the following holds for project p_j and lecturer l_k (where l_k offers p_j):

(P1) Both p_j and l_k are undersubscribed in M.

(P2) p_j is undersubscribed, l_k is full, and $s_i \in M(l_k)$.

(P3) p_j is undersubscribed, l_k is full, and l_k prefers s_i to the worst student in $M(l_k)$.

(P4) p_j is full, and l_k prefers s_i to the worst student in $M(p_j)$.

A matching is *stable* if it admits no blocking pairs.

Intuitively, the blocking pair definition tries to capture all the different ways in which a student s_i and a lecturer l_k could both improve relative to a matching M if s_i were assigned to project p_j. For this to occur, s_i must either be unassigned in M, or must prefer p_j to their current project $M(s_i)$ (Condition S1). On the lecturer side, several situations may allow l_k to accept s_i. If both p_j and l_k are undersubscribed, then l_k can take on s_i (Condition P1). If l_k is full but s_i is already assigned in M to a project offered by l_k, then l_k will agree to this switch since the number of students assigned to l_k does not change and p_j has space for s_i (Condition P2).

If, however, l_k is full and s_i is not currently assigned in M to a project offered by l_k, then l_k cannot accept s_i without first removing one of their assigned students. Lecturer l_k would be willing to do this only if they prefer s_i to their worst assigned student, and provided that p_j also has space for s_i (Condition P3). Finally, if p_j itself is full, then l_k cannot accept s_i onto p_j without first

removing a student currently assigned to p_j. Again, l_k would only agree to this if they prefer s_i to the worst student currently assigned to p_j (Condition P4).

2.3.1.2 Example.

An example SPA-S instance is shown in Figure 4.1. Here, the set of students is $\mathcal{S} = \{s_1, s_2, \ldots, s_5\}$, the set of projects is $\mathcal{P} = \{p_1, p_2, \ldots, p_5\}$, and the set of lecturers is $\mathcal{L} = \{l_1, l_2\}$. Each student has a preference list over the projects they find acceptable. For example, s_1's preference list is p_1, p_2, and s_2's preference list is p_2, p_3. Also, lecturer l_1 offers p_1, p_2, p_5, while lecturer l_2 offers p_3, p_4. Each lecturer ranks students in order of preference. In this example, l_1's preference list is s_4, s_5, s_3, s_1, s_2, and the projected preference list of l_1 for p_1 includes s_3, s_1, ranked in that order.

Students' preferences	**Lecturers' preferences**	**Offers**
s_1: p_1 p_2	l_1: s_4 s_5 s_3 s_1 s_2	p_1, p_2, p_5
s_2: p_2 p_3	l_2: s_2 s_3 s_5 s_4	p_3, p_4
s_3: p_3 p_1		
s_4: p_4 p_5		
s_5: p_5 p_4		
Project capacities: $c_1 = c_2 = c_3 = c_4 = c_5 = 1$		
Lecturer capacities: $d_1 = 3,\ d_2 = 2$		

Figure 2.6: An instance I_1 of SPA-S

With respect to the SPA-S instance I_1 shown in Figure 4.1, the matching $M_1 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_4), (s_5, p_5)\}$ is a stable matching, as it does not admit any blocking pair. On the other hand, the matching $M_2 = \{(s_1, p_2), (s_2, p_3), (s_3, p_1), (s_4, p_4)\}$ is not stable since s_5 is unassigned, and both p_5 and l_1 are undersubscribed.

2.3.1.3 Structural and algorithmic results for SPA-S

We note that the Hospitals/Residents problem (HR), discussed in Section 2.2, is a special case of SPA-S where each lecturer offers exactly one project, and the capacity of each project matches that of the lecturer offering it. In this setting, projects and lecturers are essentially indistinguishable. Consequently, several structural properties known for HR can be naturally generalized to SPA-S.

Similar to HR [43], Abraham *et al.* [8] proved that every SPA-S instance admits at least one stable matching, which can be found in polynomial-time, although many stable matchings may exist. They developed two polynomial-time algorithms for SPA-S: one student-oriented and one lecturer-oriented. In the student-oriented algorithm, each unassigned student who has a non-empty list applies to the first project p_j on their list, and become provisionally assigned to that project and lecturer. If p_j is oversubscribed, then l_k rejects the worst student s_r assigned to p_j.

The pair (s_r, p_j) is subsequently deleted (in the sense that p_j is deleted from s_r's list and s_r is deleted from $\mathcal{L}_k^j$). Similarly, if the lecturer l_k is oversubscribed then l_k rejects their worst assigned student s_r. Again, the pair (s_r, p_t) is deleted, where p_t was the project most recently assigned to s_r. The student-oriented algorithm produces a *student-optimal* stable matching, in which each student is assigned their best possible project among all stable matchings. The stable matching produced by the lecturer-oriented algorithm is considered *lecturer-optimal*. However, the notion of optimality for lecturers differs slightly: while it can be viewed as lecturer-optimal, this holds in a precise but somewhat weaker sense. We discuss this further in Section 4.2.1.2.

Abraham *et al.* [8] also presented a set of structural properties exhibited by SPA-S instances, known as the Unpopular Projects Theorem [121]. This theorem, formally stated in Theorem 4.2.1, is analogous to the Rural Hospitals Theorem (Theorem 2.2.1) for HR, and some of its properties generalise naturally to SPA-S. These structural results have provided the foundation for characterising the set of stable matchings. In particular, Olaosebikan [121] showed that when each student ranks only projects offered by different lecturers, the set of stable matchings forms a distributive lattice. In Chapter 4, we extend this result by proving that the lattice structure holds without this restriction, thus generalizing results from the HR model to the more complex SPA-S setting.

2.3.2 Lecturer preferences over students including ties (SPA-ST)

Abraham *et al.* [8] proposed an extension of SPA-S, where students (respectively, lecturers) may have ties in their preference lists indicating indifference between two or more projects (respectively, students). This model is known as the Student-Project Allocation problem with lecturer preferences over Students including Ties (SPA-ST). We note that SMTI is a special case of SPA-ST in which there are an equal number of projects and lecturers, each lecturer offers exactly one project, and the capacity of each project and lecturer is one. Similar to SMTI (discussed in Section 2.1.2.3), three notions of stability arise in SPA-ST, namely weak stability, strong stability, and super-stability.

Weak stability in SPA-ST is defined similarly to stability in SPA-S, and as in the SPA-S case, a weakly stable matching is guaranteed to exist in every instance of SPA-ST. Such a matching can be found by breaking ties arbitrarily in the SPA-ST instance to form a SPA-S instance, and then using the algorithm described in Section 2.3.1.3 to find either a student-optimal or lecturer-optimal stable matching [8]. However, in contrast to SPA-S, weakly stable matchings in SPA-ST may have different sizes. This leads to the problem of finding a weakly stable matching that assigns as many students as possible to projects, denoted MAX-SPA-ST. We recall that MAX-SMTI, which is a special case of MAX-SPA-ST, is NP-hard [100]; thus, it follows that MAX-SPA-ST is also NP-hard.

To cope with this, Cooper and Manlove [27] described a $\frac{3}{2}$-approximation algorithm for MAX-SPA-ST that finds a weakly stable matching of size at least two-thirds that of a maximum weakly

stable matching. Under the other two stability criteria (strong stability and super-stability), an instance of SPA-ST need not admit a stable matching. Olaosebikan and Manlove [122] presented a polynomial-time algorithm to find a strongly stable matching or to report that none exists. The same authors described a polynomial-time algorithm for finding a super-stable matching or reporting that no such matching exists [123].

2.3.3 Lecturer preferences over projects (SPA-P)

The variant of SPA where lecturers provide preferences over the projects that they offer is known as *Student-Project Allocation problem with lecturer preferences over Projects (*SPA-P*)* [95, 102, 124]. Manlove and O'Malley [102] showed that, in a given instance of SPA-P, stable matchings can have different sizes. This has motivated the problem of finding a stable matching of maximum size, known as MAX-SPA-P, which is NP-hard even when every project and lecturer has capacity one [102]. The authors provided a 2-approximation algorithm, and subsequently Iwama *et al.* [74] presented an improved $\frac{3}{2}$ approximation algorithm, and showed that MAX-SPA-P is not approximable within a factor of $\frac{21}{19}$.

Manlove *et al.* [95] later described an Integer Programming model for MAX-SPA-P and showed that the problem remains NP-hard even when there are only two lecturers, but becomes tractable when there is a single lecturer [96]. Furthermore, $(3, 3)$-MAX-SPA-P, where each preference list has length at most three, is also NP-hard. A different notion of stability, known as *strong stability*, was introduced by O'Malley [124], who also presented a polynomial-time algorithm for finding a strongly stable matching in SPA-P or reporting that none exists.

2.4 Related SPA models

Fleiner [39, 40] recently introduced a matroid framework for analysing stable matchings in bipartite matching problems. In this formulation, stable matchings are characterised as matroid kernels. The SPA-S model can be embedded into this framework by viewing the set of students as a partition matroid and representing lecturers using a truncation of a direct sum of uniform matroids, as noted in [8]. In this representation, the bipartite graph is modelled as a multigraph, with vertices on one side corresponding to students, vertices on the other side to lecturers, and edges representing acceptable student–project pairs.

A structural property in Fleiner's framework extends the well-known Rural Hospitals Theorem: if the matroid kernel for a particular stable matching does not span the entire ground set, then exactly the same subset is spanned in every stable matching. This generalises the property that any agent who is undersubscribed in one stable matching will be assigned the same set of partners across all stable matchings. However, this property does not hold in SPA-S, where a project or lecturer might be undersubscribed in one stable matching but receive different sets of students in

another. This observation suggests that although SPA-S can be embedded into Fleiner's matroid framework, certain structural and optimality properties do not necessarily carry over.

The Laminar Classified Stable Matching (LCSM) model introduced by Huang [60] is a generalization of HR, consisting of institutes and applicants who have preferences over one another. Each institute classifies applicants into a laminar family of classes and specifies upper and lower bounds on the number of applicants it can accept from each class. The authors noted that LCSM reduces to a special case of SPA-S when the classifications form simple partitions and no lower bounds are imposed. We note that LCSM does not capture the cases in SPA-S where a student may be assigned to different projects offered by the same lecturer across different stable matchings.

Chapter 3

Complexity Results for Restricted Variants of SPA

3.1 Introduction

A common approach to addressing the computational hardness of matching problems is to consider restricted versions of the input instances under which the problem becomes tractable, or to explore whether efficient algorithms can be developed by focusing on specific structural parameters. In this chapter, we explore this idea in the context of the Student–Project Allocation problem, focusing on two variants: SPA-P, where both students and lecturers have preferences over the projects they offer, and SPA-ST, where students have preferences over projects, lecturers have preferences over students who find their projects acceptable, and ties may occur in both students' and lecturers' preference lists. Both variants were introduced in Sections 2.3.2 and 2.3.3.

We recall that an instance of SPA-P may admit stable matchings of different sizes, while an instance of SPA-ST may admit weakly stable matchings of different sizes. Moreover, the problem of finding a stable matching of maximum size in both models is NP-hard [100, 102]. Our goal is to understand how natural restrictions on the input, such as placing bounds on the ordering of agents' preference lists, or exploiting specific structural parameters, influence the complexity of finding a maximum-size stable matching. In both problems, we consider restrictions on the number of lecturers involved and the use of a master list of projects. These restrictions are motivated by real-world matching schemes, where practical considerations may limit the way preferences are expressed.

For example, the use of master lists has featured in applications such as the Medical Training Application Service (MTAS) for allocating junior doctors to medical posts in the UK [125]. In this setting, applicants were assigned numerical scores based on their academic records and application forms, and a master list containing ties was derived from these scores. There are

also settings where the presence of a master list simplifies the problem. The variant of MAX-SMTI in which all ties are on one side and occur at the ends of preference lists is known to be NP-hard [100]. However, if there is a master list on one side, with a single tie at the tail of the list, and all preferences on the other side are strict (with or without a master list), then a maximum-size weakly stable matching can be found in polynomial time. This is an example where the presence of a master list makes the problem easier.

We then consider SPA-P from the perspective of parameterised complexity. Fixed-parameter tractable algorithms have been developed for several NP-hard variants of matching problems, including MAX-SMTI and MAX-HRT, but no such results are currently known for SPA-P. Here, we consider a new variant of SPA-P, where we introduce a parameter, *project topics*, such that each project belongs to a single topic. This parameter may arise naturally in university settings, where multiple projects fall under a common topic, and students are indifferent between projects within the same topic. As a result, students express strict preferences over topics rather than over individual projects. In this context, the project's topic serves as a particularly meaningful parameter, as it reflects how projects are grouped in practice and how both students and lecturers may structure their preferences over such projects in real academic settings.

3.1.1 Background and motivation

As noted in Section 2.1.2.3, computing a maximum-size weakly stable matching in SMTI is NP-hard [100], with similar results holding for HRT [102] and SPA-ST [96]. These problems remain intractable even under strong restrictions, such as bounded preference list length or when each tie is of length at most two [68, 71, 100, 137]. In contrast, several stable matching problems in the SMT setting, such as generating weakly stable matchings, identifying all weakly stable pairs, or computing an egalitarian matchings, are solvable in polynomial time. Moreover, when agents follow a master list, these problems often admit faster or simpler algorithms than in the general case.

We recall that MAX-SPA-ST is the problem of computing a weakly stable matching of maximum size in a given instance of SPA-ST. Moreover, this problem is NP-hard. This follows from the fact that MAX-SMTI is NP-hard, and since SMTI is a special case of SPA-ST, this implies the NP-hardness of MAX-SPA-ST. Similarly, the problem of finding a maximum-size stable matching, known as MAX-SPA-P, is also NP-hard [102]. This result holds even under strong restrictions, such as when only two lecturers are involved or when all preference lists have length at most three [96]. Furthermore, the best known approximation algorithm for MAX-SPA-P achieves a performance guarantee of $\frac{3}{2}$ [74].

These complexity results motivate the study of restricted variants of the problem, with the aim of identifying the boundary between tractable and intractable cases. For example, bounding the lengths of preference lists has been shown to yield efficient algorithms for MAX-SMTI [68],

illustrating that certain restrictions on the input can significantly influence the computational complexity. In this chapter, we examine the complexity of MAX-SPA-P and MAX-SPA-ST under similar restrictions. As we will show, even seemingly simple constraints can determine whether a problem is solvable in polynomial time or remains NP-hard.

3.1.2 Contributions and structure of the chapter

The results presented in this chapter are grouped into two parts. The first part focuses on classical complexity results. In Section 3.2, we formally define SPA-ST and in Section 3.2.2, we prove that computing a maximum size weakly stable matching, known as MAX-SPA-ST, remains NP-hard even when the instance contains only a single lecturer. We then turn our attention to the SPA-P setting in Section 3.3. In Section 3.3.2, we observe that MAX-SPA-P remains NP-hard even when the preference lists of both students and lecturers are derived from a master list of projects. On the positive side, in Section 3.3.3, we provide a polynomial-time algorithm for MAX-SPA-P-SL. In this setting, each student finds acceptable only the projects offered by a single lecturer. Finally, in Section 3.3.4, we show that when all students have identical preferences over projects, MAX-SPA-P is also solvable in polynomial time.

The second part, which forms the main contribution of this chapter, focuses on the parameterised complexity of MAX-SPA-P. In Section 3.4.1, we provide a brief background on known FPT results for stable matching problems. In Section 3.4.2, we present a new variant of SPA-P in which we introduce a natural parameter, *project topics*, such that both students and lecturers express strict preferences over project topics but are indifferent between projects belonging to the same topic. As a further restriction, we impose *uniform capacities*, meaning that each lecturer and each project they offer have the same capacity. We refer to this variant as SPA-P with uniform capacities, abbreviated SPA-PUC.

We note that computing a maximum-size stable matching in SPA-PUC is NP-hard, even under uniform capacities, since MAX-SPA-P is NP-hard even when each lecturer and project capacity is one. Also, given individual preference lists, we can first derive preferences over project topics and then compute, based on these, a partition of students and lecturers into types that satisfies the definition of a typed instance. In Sections 3.4.4 and 3.4.5, we prove that finding a maximum-size stable matching in SPA-PUC is fixed-parameter tractable when parameterized by the number of project topics. The result follows from an Integer Linear Programming (ILP) formulation whose number of variables depends only on the number of topics. This provides a positive result for a subclass of SPA-P instances, particularly when the number of project topics is small.

3.2 Complexity result for SPA-ST under weak stability

In this section, we focus on the SPA-ST problem. We begin by formally defining SPA-ST in Section 3.2.1, before presenting our results in Section 3.2.2. All notation and terminology introduced for SPA-S in Section 2.3.1 extend naturally to SPA-ST, with the key distinction that preference lists of students and lecturers in SPA-ST may contain ties.

3.2.1 Formal definition of SPA-ST

Formally, an instance I of SPA-ST consists of three sets: a set of students $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_1}\}$, a set of projects $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_2}\}$, and a set of lecturers $\mathcal{L} = \{l_1, l_2, \ldots, l_{n_3}\}$. A pair (s_i, p_j) is *acceptable* if project p_j appears on student s_i's preference list and s_i appears on the preference list of the lecturer l_k, who offers p_j. As in SMTI, three notions of stability arise when preferences include ties: weak stability, strong stability, and super-stability [54, 94, 122, 123]. In this chapter, we focus on *weak stability*. We note that weak stability in SPA-ST is defined in the same way as stability in SPA-S, which we restate as follows:

Definition 3.2.1 (Weak stability in SPA-ST)**.** Let I be an instance of SPA-ST and M a matching in I. An acceptable pair $(s_i, p_j) \notin M$ is a *blocking pair* for M if the following conditions hold:

(S1) s_i is unassigned in M, or

(S2) s_i is assigned in M but prefers p_j to their assigned project $M(s_i)$,

and one of the following holds for project p_j and lecturer l_k offering p_j:

(P1) Both p_j and l_k are undersubscribed in M.

(P2) p_j is undersubscribed in M, l_k is full in M, and $s_i \in M(l_k)$

(P3) p_j is undersubscribed in M, l_k is full in M, and l_k prefers s_i to the worst student in $M(l_k)$.

(P4) p_j is full in M, and l_k prefers s_i to the worst student in $M(p_j)$.

To illustrate that weakly stable matchings in SPA-ST may differ in size, consider the instance I_2 shown in Figure 3.1, which involves three students, three projects, and two lecturers. Student s_1 is indifferent between projects p_3 and p_2 (indifference is indicated by round brackets in the preference lists). This instance admits two weakly stable matchings of different sizes: $M_1 = \{(s_1, p_3), (s_3, p_2)\}$ and $M_2 = \{(s_1, p_2), (s_2, p_3), (s_3, p_1)\}$.

Students' preference	Lecturers' preference	offers
s_1: $(p_3 \quad p_2)$	l_1: $s_1 \; s_3$	p_1, p_2
s_2: p_3	l_2: $s_1 \quad s_2 \quad s_3$	p_3
s_3: $p_3 \quad p_2 \quad p_1$		
Project capacities:	$c_1 = 2;\ c_2 = c_3 = 1$	
Lecturer capacities:	$d_1 = 2,\ d_2 = 1$	

Figure 3.1: An instance I_2 of SPA-ST.

3.2.2 Complexity of MAX-SPA-ST with one lecturer

In this section, we consider the complexity of MAX-SPA-ST, under the restriction that there is only one lecturer in the instance. Let I be an instance of SPA-ST with one lecturer, and let $s^+(I)$ denote the size of a maximum size stable matching in I. We show that finding a maximum size stable matching in SPA-ST even under this restriction is NP-complete by presenting a polynomial-time reduction from a restricted variant of the Stable Marriage problem with Ties and Incomplete Lists (SMTI) involving master lists on both sides. This variant, known as COMPLETE SMTI-2ML, is defined next.

3.2.2.1 COMPLETE SMTI-2ML

In SMTI, each man and woman may omit certain partners off their preference lists (incomplete lists) and may be indifferent between two or more acceptable partners (ties). A matching is said to be *weakly stable* if it admits no blocking pair in which both agents prefer each other to their current partners. The variant SMTI-2ML assumes that preference lists are derived from *master lists* on both sides, i.e., a master list of men from which the women's preferences are derived, and a master list of women from which the men's preferences are derived. A *master list of men* is a single list containing all men, possibly with ties. Each woman's preference list contains her acceptable partners ranked precisely according to the master list. Thus, the preference list of each woman w follow the master list exactly, except that each man m that w finds unacceptable is deleted. A *master list of women* is defined analogously.

Let MAX SMTI-2ML be the problem of computing a maximum size weakly stable matching in an instance of SMTI-2ML. The corresponding decision problem is MAX SMTI-2ML-D, which asks whether there exists a weakly stable matching of size at least k, for a given integer k. Irving et al. [71] showed that MAX SMTI-2ML-D is NP-complete, even under various restrictions on the positions and lengths of ties in the master lists, as well as on the lengths of individual preference lists. We focus on a special case, referred to as COMPLETE SMTI-2ML, in which the number of men equals the number of women, and the target size of the matching is exactly n, where n is the number of men (and women). Finding a weakly stable matching of size n, that is a *complete weakly stable matching*, is NP-complete [71].

Theorem 3.2.1 ([71]). COMPLETE SMTI-2ML *is* NP*-complete:*

Name: COMPLETE SMTI-2ML
Instance: *An instance of* SMTI-2ML *with* n *men and* n *women.*
Question: *Does the instance admit a weakly stable matching of size* n*?*

3.2.2.2 MAX-SPA-ST with one lecturer

We prove that this problem is NP-complete by reducing from the known NP-complete problem COMPLETE SMTI-2ML. To do so, we first define the following decision problem:

Name: MAX-SPA-ST with one lecturer
Instance: An instance I of SPA-ST with one lecturer, and an integer $k \in \mathbb{Z}^+$.
Question: Does there exist a stable matching M in I such that $|M| \geq k$?

Theorem 3.2.2. MAX-SPA-ST *is* NP*-complete even when there is only one lecturer involved. The result holds even if each project has capacity* 1.

We first note that MAX-SPA-ST with one lecturer is in NP, since given a matching M, we can verify in polynomial time whether M is stable and whether it has size at least k. Let I' be an instance of COMPLETE SMTI-2ML, consisting of n men and n women. We construct a corresponding SPA-ST instance I as follows.

For each man m_i in I', introduce a student s_i in I. For each woman w_j, create a project p_j with capacity 1. All projects are offered by a single lecturer l, who has capacity n. The lecturer's preference list is derived from the master list of men in I': for each man m_i, if m_i appears (possibly in a tie), the corresponding student s_i is added in the same position, preserving any ties. Each student s_i's preference list is identical to the preference list of the corresponding man m_i, replacing each woman w_j with the corresponding project p_j.

We set the target size k in the constructed instance I to be n, where n is the number of men and women in the original COMPLETE SMTI-2ML instance I'. That is, we ask whether I admits a weakly stable matching of size at least $k = n$. We claim that I' admits a complete weakly stable matching of size n if and only if I admits a weakly stable matching of size k.

Lemma 3.2.1. *If* I' *admits a complete weakly stable matching* M' *of size* n*, then* I *admits a weakly stable matching* M *of size at least* k*, where* $k = n$*.*

Proof. Suppose I' admits a complete weakly stable matching M' of size n. We construct a matching M in I by assigning s_i to p_j whenever m_i is assigned to w_j in M'. Since there are exactly n

men and women in I' and $|M'| = n$, all students in I are assigned, and thus lecturer l, who offers all projects, is assigned exactly n students. Clearly, M is a valid matching in I, since each student is assigned to at most one project, each project has capacity one and is not oversubscribed, and the lecturer, who offers all projects, is assigned no more than n students in total.

Now suppose, for contradiction, that M admits a blocking pair (s_i, p_j) in I. Then either s_i is unassigned in M or s_i prefers p_j to $M(s_i)$, and one of the following conditions holds:

(a) both p_j and l are undersubscribed in M.

(b) p_j is undersubscribed in M, l is full in M, and $s_i \in M(l_k)$

(c) p_j is undersubscribed in M, l is full in M, and l prefers s_i to the worst student in $M(l)$.

(d) p_j is full in M, and l prefers s_i to the worst student in $M(p_j)$.

By construction of M, all n students in I are assigned to n different projects. Moreover, each project has capacity one and is offered by the single lecturer l, whose total capacity is also n. Therefore, both p_j and l are full in M, ruling out cases (a), (b), and (c). Case (d) implies that l prefers s_i to the worst student in $M(p_j)$, say s_t. Moreover, $(s_i, p_j) \notin M$. But by construction, this would translate to a pair $(m_i, w_j) \notin M'$ who would prefer to be assigned to each other than their assigned partners in I'. However, (m_i, w_j) would block M', a contradiction. Therefore, M is stable. Finally, by construction, $|M| = |M'| = k = n$, and so M is a weakly stable matching in I of size n. □

Lemma 3.2.2. *If I admits a weakly stable matching M of size at least k, then I' admits a complete weakly stable matching M' of size n.*

Proof. Suppose that I admits a weakly stable matching M of size k. We construct a matching M' in I' by including the pair (m_i, w_j) in M' for each $(s_i, p_j) \in M$. This yields a valid matching in I': since no student is multiply assigned in M, it follows that no man is assigned to more than one woman in M'.

We now show that M' is a complete weakly stable matching. Suppose for contradiction that there exists a pair (m_i, w_j) that blocks M' in I'. Then m_i is either unassigned in M' or prefers w_j to $M'(m_i)$, and w_j is either unassigned or prefers m_i to $M'(w_j)$. If both m_i and w_j are unassigned in M', this would mean that the corresponding student s_i is unassigned in M, and both p_j and l are undersubscribed in M. This yields a blocking pair in M, a contradiction.

Otherwise, suppose m_i prefers w_j to his assigned partner $M'(m_i)$ and w_j is assigned to some man m_r. Then the corresponding student s_i prefers p_j to $M(s_i)$, and p_j is full with some student s_r. Since all projects have capacity one and are offered by the single lecturer l, and since all

n students are assigned in M, l must be full. Hence, l prefers s_i to s_r, and (s_i, p_j) blocks M, a contradiction. Therefore, no such blocking pair exists and M' is a complete weakly stable matching in I'. Moreover, since M has size n and each assigned pair in M corresponds to a unique pair in M', we have $|M'| = |M| = n$, completing the proof. □

By Lemmas 3.2.1 and 3.2.2, we have shown that I' admits a complete weakly stable matching of size n if and only if I has a weakly stable matching M of size at least k. This completes the proof.

3.3 Complexity results for SPA-P

In this section, we examine how restrictions on preference lists influence the complexity of MAX-SPA-P. We begin in Section 3.3.1 by formally defining the SPA-P model. In Section 3.3.2, we consider the variant of SPA-P in which preference lists are derived from a master list of projects. In Section 3.3.3, we study another case, denoted SPA-P-SL, where each student ranks only projects offered by the same lecturer, and present a polynomial-time algorithm for computing a maximum stable matching in SPA-P-SL.

3.3.1 Formal definition of SPA-P

Formally, an instance I of SPA-P consists of a set $\mathcal{S} = \{s_1, \ldots, s_{n_1}\}$ of students, a set $\mathcal{P} = \{p_1, \ldots, p_{n_2}\}$ of projects, and a set $\mathcal{L} = \{l_1, \ldots, l_{n_3}\}$ of lecturers. Unlike SPA-S, lecturers rank their offered projects in strict order instead of ranking students. All notation and terminology from SPA-S apply, except for stability, which we define next.

Definition 3.3.1 (Stability in SPA-P [102])**.** Let I be an instance of SPA-P and M a matching in I. An acceptable pair $(s_i, p_j) \notin M$ is a *blocking pair* for M if p_j is undersubscribed in M and the following conditions hold for both student s_i and lecturer l_k, who offers p_j:

(S1) s_i is either unassigned in M, or

(S2) s_i prefers p_j to their assigned project $M(s_i)$.

and one of the following holds for lecturer l_k:

(L1) $s_i \in M(l_k)$ and l_k prefers p_j to $M(s_i)$.

(L2) $s_i \notin M(l_k)$ and l_k is undersubscribed in M.

(L3) $s_i \notin M(l_k)$ and l_k prefers p_j to their worst non-empty project in M.

A matching may also be undermined by a group of students acting together, forming a *coalition*. Formally, given a matching M, a coalition is a sequence of students $C = \langle s_{i_0}, \ldots, s_{i_{r-1}} \rangle$ for some

$r \geq 2$, where each student s_{i_j} (for $0 \leq j \leq r-1$) is assigned in M and prefers the project assigned to $s_{i_{j+1}}$ over their own assignment $M(s_{i_j})$, with addition modulo r. If all students in the coalition simultaneously swap to the project of the next student in the sequence, each becomes strictly better off, while the overall size of the matching remains unchanged. Moreover, since all lecturers are assumed to be indifferent between students assigned to their projects, no lecturer is worse off. A matching is said to be *coalition-free* if it admits no such coalition. We define a matching M to be *stable* if it admits no blocking pairs and it is coalition-free.

For example, in the SPA-P instance I_3 shown in Figure 3.2, the matching $M_1 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3)\}$ admits a coalition $\{s_1, s_2\}$, since both students prefer each other's assigned project to their own. By swapping their assigned projects, we obtain $M_2 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$, in which both s_1 and s_2 receive a more preferred project. For instance, the matchings $M_2 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ and $M_3 = \{(s_1, p_3), (s_2, p_1)\}$ admit no blocking pairs and are coalition-free. Hence, they are both stable in the instance shown in Figure 3.2.

Students' preferences	Lecturers' preferences
s_1: p_3 p_2 p_1	l_1: p_2 p_1
s_2: p_1 p_2	l_2: p_3
s_3: p_3	
	Project capacities: $c_1 = c_2 = c_3 = 1$
	Lecturer capacities: $d_1 = 2,\ d_2 = 1$

Figure 3.2: An instance I_3 of SPA-P.

3.3.2 SPA-P with master lists

Here, we consider the problem of finding a maximum stable matching in an instance of SPA-P where master preference lists are imposed. In this setting, a master list is a global ranking of projects from which each student and lecturer derives their individual preference lists. Manlove and O'Malley [102] proved the NP-hardness of MAX-SPA-P via a reduction from the problem of finding a minimum maximal matching (MIN MM), by constructing the instance I' of SPA-P shown in Figure 3.3, where each project and lecturer has capacity 1.

Students' preferences		Lecturers' preferences	
u_i^1: r_i p_{j_i} p_{k_i} t_i	$(1 \leq j \leq n_1)$	w_j: p_j q_j	$(1 \leq j \leq n_2)$
u_i^2: r_i p_{k_i} p_{j_i}	$(1 \leq j \leq n_1)$	x_j: r_j	$(1 \leq j \leq n_1)$
s_i: q_i	$(1 \leq j \leq n_2)$	y_j: t_j	$(1 \leq j \leq n_1)$

Figure 3.3: Preference lists for constructed instance of SPA-P due to [102]

We note that MAX-SPA-P with a master list of projects is NP-hard, since the reduction used to prove

the hardness of MAX-SPA-P still applies in this restricted setting, as illustrated in Figure 3.3. In particular, the construction admits a natural master list of projects: $(r_1, r_2, \ldots, r_{n_1}, p_1, p_2, \ldots, p_{n_1}, t_1, t_2, \ldots, t_{n_1}, q_1, q_2, \ldots, q_{n_2})$. Since the preferences of all agents can be derived from this list, the reduction applies in the master list setting, and the NP-hardness result follows.

3.3.3 SPA-P with projects offered by the same lecturer

We now consider a restricted version of SPA-P, denoted SPA-P-SL, in which each student ranks only projects offered by the same lecturer. We define MAX-SPA-P-SL as the problem of finding a maximum size stable matching in an instance of SPA-P-SL. To solve this problem, we exploit the structure of the input by dividing it into independent sub-instances, one for each lecturer. Each sub-instance consists of the set of students whose preference lists include only projects offered by a particular lecturer, along with those projects and the lecturer who offers them. These sub-instances fall into the special case MAX-SPA-P-L1, where all projects in the instance are offered by a single lecturer. Recall that a polynomial-time algorithm for MAX-SPA-P-L1 is described in [96], and so by solving each sub-instance independently using this algorithm and combining the resulting stable matchings, we obtain an optimal solution for MAX-SPA-P-SL.

3.3.3.1 Polynomial-time algorithm for MAX-SPA-P-SL

Let I be an instance of MAX-SPA-P-SL involving n_1 students and n_2 lecturers. We assume standard notation and terminology from the general SPA-P setting. In this restricted variant, each student ranks only projects that are offered by a single lecturer. To compute a maximum stable matching in I, we construct a collection of sub-instances $I_1, I_2, \ldots, I_{n_2}$, where each sub-instance I_k corresponds to a lecturer l_k in the original instance.

For each lecturer l_k, we identify the set S_k of students who find acceptable at least one project offered by l_k. The sub-instance I_k consists of these students S_k, the projects they find acceptable that are offered by l_k, and the lecturer l_k, all with their original preference lists and capacities. Since I_k is an instance of SPA-P involving only one lecturer, we apply the polynomial-time algorithm for MAX-SPA-P-L1 given in [96] to compute a maximum stable matching M_k for each I_k. By construction, each student appears in at most one sub-instance, so matchings $M_1, M_2, \ldots, M_{n_2}$ are disjoint and can be combined to yield a stable matching M in the original instance I. The full procedure is described formally in Algorithm 1.

Theorem 3.3.1. *Let I be an instance of* SPA-P-SL *with k lecturers, n_1 students, and total preference list length l. Let R denote the maximum rank of any project on a student's preference list. Then Algorithm* MAX-SPA-P-SL *computes a stable matching of maximum size in time $O(kn_1^2Rl)$.*

Algorithm 1 MAX-SPA-P-SL-$\mathcal{S}$

Input: An instance I of SPA-P-SL
Output: A maximum-size stable matching M in I
Initialize $M \leftarrow \emptyset$
Initialize an empty list of sub-instances $\mathcal{I}$
for each lecturer l_k in I **do**
    Let P_k be the set of projects offered by l_k
    Let S_k be the set of students who find some project in P_k acceptable
    Construct a sub-instance I_k involving S_k, P_k, and l_k
    Add I_k to $\mathcal{I}$
end for
for each sub-instance I_k in $\mathcal{I}$ **do**
    Run the MAX-SPA-P-L1 algorithm on I_k to compute a stable matching M_k
    Add all pairs in M_k to M
end for
return M

Proof. We first prove that M is stable. Suppose for contradiction that M admits a blocking pair (s_i, p_j). Now suppose the sub-instance s_i belongs to is I_k and the maximum-size stable matching produced during an execution E of the algorithm is M_k. By construction of M, (s_i, p_j) being a blocking pair in M implies (s_i, p_j) is also a blocking pair in the matching M_k. However, this contradicts the fact that M_k is a stable matching in I_k. Hence, M is stable. Suppose M is not the maximum. Then there exists another stable matching M^* such that $|M^*| > |M|$. Suppose that $(s_i, p_j) \in M^* \setminus M$, and (s_i, p_j) belongs to a sub-instance I_k whose stable matching output is M_k. This implies that the size of the stable matching M_k can be increased by $\{M_k \cup (s_i, p_j)\}$, a contradiction to the fact that M_k is a maximum stable matching. We can also verify that M is coalition-free since we have a different set of students in each I_k, and the matching M_k obtained in each I_k is coalition-free. Thus, M is a maximum-size stable matching.

The partitioning step of the algorithm requires $O(kn_1R)$ time, since for each of the k lecturers, we examine all n_1 students with preference lists of length at most R. Running the MAX-SPA-P-L1 algorithm on each sub-instance has a time complexity of $O(kn_1^2Rl)$, where l is the total length of all student preference lists. Hence, the overall time complexity of the algorithm is $O(kn_1^2Rl)$. □

3.3.4 Students with identical preferences

Here, we consider a restriction of SPA-P where all students have identical preferences over projects, denoted (1,TYPE)-SPA-P. We prove that the problem of finding a maximum-size stable matching in this setting is solvable in polynomial time. In particular, Lemma 3.3.1 shows that every stable matching in a given instance I of (1,TYPE)-SPA-P assigns the same number of students. Hence, it follows that any stable matching admitted by I is of maximum size. Consequently, any algorithm for computing a stable matching in the general SPA-P setting can be applied here. In particular, the 2-approximation algorithm for MAX-SPA-P proposed by Manlove and O'Malley [102] returns a

stable matching in $O(\lambda)$ time, where λ is the length of a student's preference list. This algorithm yields an optimal solution in this setting.

Lemma 3.3.1. *Given an instance I of* (1,TYPE)-SPA-P*, the same number of students are assigned in every stable matching.*

Proof. Let I be an instance of (1,TYPE)-SPA-P, and let M and M' be any two arbitrary stable matchings in I. Suppose for a contradiction that $|M'| > |M|$. Then there exists some student s who is assigned in M' but not in M. Moreover, there is some lecturer l_k such that $|M'(l_k)| > |M(l_k)|$. Furthermore, there must be some project $p_j \in P_k$ such that $|M'(p_j)| > |M(p_j)|$. Hence both l_k and p_j are undersubscribed in M. However, since each student in I have identical preferences and some student is assigned to p_j in M', then s also finds p_j acceptable. Therefore (s, p_j) blocks M, a contradiction. Thus, $|M'| = |M|$ since M and M' are two arbitrary stable matchings. □

3.4 Parameterised complexity of SPA-P

Parameterized complexity provides a way to cope with NP-hard problems by confining the exponential part of the running time to one or more parameters of the input. The running time of a parameterised algorithm is expressed as a function of both the parameter k and the overall input size n, typically written as $f(k) \cdot n^{O(1)}$, where f is a computable function depending only on k. When k is small in practice, this approach can lead to efficient algorithms even for large input instances.

In this section, we explore this idea in the context of the Student–Project Allocation problem with lecturer preferences over Projects (SPA-P). We introduce a natural parameter: *project topics*, where each project offered by a lecturer belongs to a single topic. This parameter arises naturally in university settings, where projects can be grouped by research themes or subject areas, and students are usually interested in the broader topic rather than in specific individual projects. In these situations, it is reasonable to assume that students express strict preferences over topics and are indifferent between projects within the same topic.

3.4.1 Parameterised stable matching problems

The parameterized complexity of several NP-hard variants of stable matching problems has recently been studied, with the aim of identifying parameters that enable efficient algorithms. We recall the brief discussion on FPT in Section 1.1.2.3. In this section, we present results in the bipartite matching setting, focusing in particular on problems that involve ties and incomplete preference lists.

Adil *et al.* [9] showed that the problem of computing a maximum stable matching in SMTI is FPT when parameterised by the size of the matching. Marx and Schlotter [106] later proved that the problem is fixed parameter tractable when parameterised by the total length of ties across all preference lists. However, they also showed that the problem becomes W[1]-hard when parameterised by the number of ties in the instance, even if all the men have strictly ordered preference lists. Gupta *et al.* [51] considered structural restrictions on the *acceptability graph* of an instance, where agents are represented as vertices and an edge connects two agents if they find each other acceptable. They proved that the problem remains W[1]-hard when parameterised by the treewidth of this acceptability graph.

Boehmer *et al.* [19] studied the *Incremental Stable Marriage* problem, in which an instance of the Stable Marriage problem is subject to modifications through changes in the agents' preference lists. Given a matching that was stable in the original instance, the goal is to compute a new matching that is stable with respect to the modified preferences and remains as close as possible to the original matching. They show that the problem is W[1]-hard when parameterized by the number of ties introduced by the changes, but also identify tractable cases, including fixed-parameter algorithms and polynomial-time results when the number of distinct preference lists is small.

An alternative parameterisation, introduced by Meeks and Rastegari [114], considers instances in which agents can be partitioned into a bounded number of *types*. Each agent's preference list is defined over types rather than over individuals, and agents of the same type have identical preferences over types. Moreover, each agent is indifferent between all acceptable candidates of a given type. Under this assumption, they showed that MAX-SMTI is in FPT when parameterised by the number of types. A similar result holds for MAX-HRT. They also considered two generalisations. In the first, agents of the same type may have different preference lists, provided that all candidates of the same type appear in a single contiguous block in each list. Under this condition, TYPED MAX-SMTI and TYPED MAX-HRT are fixed-parameter tractable with respect to the number of types. If, in addition, preferences over types are strict, the problem becomes polynomial-time solvable.

In the second generalisation, each agent may include a small number of *exceptional candidates*, meaning specific individuals who are ranked explicitly rather than according to their type. If each agent includes at most one exceptional candidate at the top of their list, TYPED MAX-SMTI is in FPT. However, if agents are allowed to include two or more exceptional candidates in arbitrary positions on their preference list, the problem becomes NP-hard, even when the number of types is bounded by a constant. In this case, the problem is not in XP unless P= NP.

3.4.2 SPA-P with uniform capacities

In this section, we consider SPA-PUC, a restricted version of SPA-P that incorporates project topics and enforces uniform capacities. In this model, each lecturer offers a set of projects, and both the lecturer and the projects they offer have the same capacity. Similar to the general case, an instance of SPA-PUC may admit stable matchings of different sizes, and we are interested in finding one of maximum size.

Project topics. In many real-world settings, students may care more about the general subject area of a project than about the individual project itself. For example, a student might be interested in Algorithmics and be willing to work on any project in that area, without having a preference between the specific projects it includes. However, they may still prefer projects in Algorithmics overall to those in another area, such as Robotics. This kind of scenario motivates us to consider a version of SPA-P in which projects are classified according to project topics.

In SPA-PUC, students express strict preferences over project topics rather than individual projects, and are indifferent between any two projects that belong to the same topic. Specifically, if a student finds one project in a given topic acceptable, then all projects in that topic are considered equally acceptable. Moreover, students may be partitioned into *types* based on their preferences over topics. A student is of *type* i if their list of topics is ordered in exactly the same way as that of all other type-i students. We use $t(p_j)$ to denote the topic of project p_j. A student s_i is said to *strictly prefer* project p_a to project p_b if they prefer $t(p_a)$ to the topic of $t(p_b)$; a similar definition holds for each lecturer. For any two topics $t(a)$ and $t(b)$, we write $t(a) \succeq_i t(b)$ to mean that topic $t(a)$ is preferred at least as much as topic $t(b)$ by type-i students, and $t(a) \succ_i t(b)$ if topic $t(a)$ is strictly preferred to topic $t(b)$ by type-i students.

Lecturers also express preferences over topics rather than individual projects, and are indifferent between any two projects in the same topic. Two lecturers are of the same type if they have identical preferences over topics. A lecturer may offer projects from multiple topics, and is not required to offer all projects under any given topic. For any lecturer l_k, we write $t(a) \succeq_k t(b)$ to mean that topic $t(a)$ is preferred at least as much as topic $t(b)$ by l_k, and $t(a) \succ_k t(b)$ if topic $t(a)$ is strictly preferred to topic $t(b)$.

Uniform capacities. Each lecturer l_k offers a set of projects P_k, and every project $p_j \in P_k$ has the same capacity as the lecturer; that is, $c_j = d_k$.

Example: As an example, consider the instance shown in Figure 3.4. There are two project topics: topic 1 includes p_1, p_2, p_5, and topic 2 includes p_3, p_4, p_6. Students s_1 and s_3 are of the same type, as they both strictly prefer topic 1 over topic 2. In contrast, student s_2 is of a different type, as she is indifferent between the two topics. On the lecturer side, l_1 and l_3 are of the same type, since they offer and strictly prefer projects in topic 1 to those in topic 2. Lecturer l_2 is of

a different type, as they only offer (and prefer) a project in topic 2. Moreover, lecturer l_1 offers projects $p_1, p_2,$ and p_6, and both the lecturer and each of these projects have capacity 2. The same applies to every lecturer and the projects they offer.

Students' preferences	Lecturers' preferences
s_1: $(p_1 \quad p_2 \quad p_5)$ $(p_3 \quad p_4 \quad p_6)$	l_1: $(p_1 \quad p_2)$ $\quad p_6$
s_2: $(p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6)$	l_2: $\quad p_3$
s_3: $(p_1 \quad p_2 \quad p_5)$ $(p_3 \quad p_4 \quad p_6)$	l_3: $\quad p_5 \quad p_4$
	Project capacities: $c_1 = c_2 = c_6 = 2; c_3 = c_4 = c_5 = 1$
	Lecturer capacities: $d_1 = 2;\ d_2 = d_3 = 1$

Figure 3.4: An instance I_1 of SPA-PUC

We now give a revised definition of a blocking pair in the SPA-PUC setting. This definition is similar to that in the general SPA-P model, except that preferences are expressed over project topics rather than individual projects.

Definition 3.4.1 (Blocking Pair in SPA-PUC)**.** Let I be an instance of SPA-PUC, and let M be a matching in I. An acceptable pair $(s_i, p_j) \in (\mathcal{S} \times \mathcal{P}) \setminus M$ is a *blocking pair* for M if p_j is undersubscribed in M and the following conditions hold for both s_i and the lecturer l_k who offers p_j:

(S1) either s_i is unassigned in M, or

(S2) s_i prefers p_j's topic to the topic of their assigned project $M(s_i)$,

and one of the following holds for l_k:

(L1) $s_i \in M(l_k)$ and l_k prefers p_j's topic to the topic of $M(s_i)$;

(L2) $s_i \notin M(l_k)$ and l_k is undersubscribed in M;

(L3) $s_i \notin M(l_k)$ and l_k prefers p_j's topic to the topic of their worst non-empty project in M.

We say that M is stable if it admits no blocking pairs.

In the general SPA-P setting, a matching is said to be *stable* if it admits no blocking pair and no coalition. A *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$, each of whom is assigned in a matching M, such that for all $j \in \{0, \ldots, r-1\}$, student s_{i_j} strictly prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is taken modulo r. In SPA-PUC, we define stability in terms of blocking

pairs only, as shown in Definition 3.4.1. We do not require that matchings are coalition-free. This simplification is justified by the fact that any coalition present in a matching can be resolved in polynomial time: once a coalition is identified, the students can cyclically swap their assigned projects so that each one is strictly better off. Moreover, such a swap does not change the number of students assigned to projects and lecturers, and does not affect the size of the matching.

3.4.3 Hardness of MAX-SPA-PUC

An instance of SPA-PUC may admit stable matchings of different sizes, and we are naturally interested in computing one of maximum size, denoted MAX-SPA-PUC. To illustrate that stable matchings in SPA-PUC can differ in size, consider the instance I_3 shown in Figure 3.5. The matchings

$$M = \{(s_1, p_1), (s_3, p_3)\} \quad \text{and} \quad M' = \{(s_1, p_3), (s_2, p_1), (s_3, p_3)\}$$

are both stable, but M has size 2 while M' has size 3.

Students' preferences	Lecturers' preferences
s_1: $(p_1 \quad p_2) \quad p_3$	l_1: $(p_1 \quad p_2)$
s_2: $(p_1 \quad p_2)$	l_2: $\quad p_3$
s_3: $\quad p_3 \quad (p_1 \quad p_2)$	
	Project capacities: $c_1 = c_2 = 1;\ c_3 = 2$
	Lecturer capacities: $d_1 = 1;\ d_2 = 2$

Figure 3.5: An instance I_3 of SPA-PUC

Although SPA-PUC includes restrictions on the capacities of projects and lecturers, we observe that MAX-SPA-PUC is also NP-hard. We recall that MAX-SPA-P is known to be NP-hard even when every project and lecturer has capacity one [102]. This case can be viewed as an instance of SPA-PUC in which each project belongs to a unique topic and all project and lecturer capacities are set to one. In this encoding, preferences over topics effectively reduce to preferences over individual projects, so the instance behaves identically to the original SPA-P setting. As a result, the NP-hardness of the general problem carries over directly to the SPA-PUC setting.

3.4.4 FPT algorithm for SPA-PUC

In this section, we show that MAX-SPA-PUC is fixed parameter tractable when parameterised by the number of project topics. We begin by showing in Lemma 3.4.1 that in order to determine whether or not a matching M is stable, it is enough to examine the number of students of each type assigned to each project. In this lemma, we provide three sufficient conditions for checking stability for some project p_j offered by l_k: (i) the number of students assigned to p_j is less than its capacity; (ii) the number of type-i students not assigned to a project at least as desirable as

$t(p_j)$ (from their own perspective) and not assigned to any project l_k ranks at least as desirable as p_j is > 0; (iii) the number of students assigned to topics that l_k ranks at least as desirable as $t(p_j)$ is less than d_k.

In Lemma 3.4.2, we show that any instance I of SPA-PUC can be transformed into an equivalent instance I' in which each lecturer offers at most one project on each topic. Finally, in Lemma 3.4.5, we prove that I can be further transformed into an equivalent instance that contains at most one lecturer of each type.

Lemma 3.4.1. *Let I be an instance of SPA-PUC, and let M be a matching in I. Suppose there are r distinct student types in I. For each student type i and each project p_j, let N_i denote the total number of type-i students in I, and let X_{ij} denote the number of type-i students assigned to p_j in M. A blocking pair exists in M if and only if there exists a type i of students and a project p_j offered by l_k where conditions (1), (2) and (3) hold as follows:*

(1) $\sum\limits_{1 \leq q \leq r} X_{qj} < c_j$

(2) $N_i - \left(\sum\limits_{t(p_m) \succeq_i t(p_j)} X_{im} + \sum\limits_{t(p_m) \succeq_k t(p_j)} X_{im} - \sum\limits_{\substack{t(p_m) \succeq_i t(p_j) \\ t(p_m) \succeq_k t(p_j)}} X_{im} \right) > 0$

(3) $\sum\limits_{\substack{t(p_m) \succeq_k t(p_j) \\ 1 \leq q \leq r}} X_{im} < d_k$

Proof. ($\Rightarrow$) First suppose that conditions (1), (2) and (3) hold. Condition (1) implies that project p_j is undersubscribed in M, since the total number of students assigned to p_j is less than its capacity. Moreover, condition (2) implies that there exists a type-i student s who is not assigned to any project whose topic is at least as desirable as $t(p_j)$ from their own perspective, and who is also not assigned to any project that l_k considers at least as desirable as p_j. Therefore, s is either unassigned or is assigned in M to a project p such that s prefers $t(p_j)$ to $t(p)$. Moreover, if $s \in M(l_k)$ then l_k finds p less desirable than p_j.

Also, by condition (3), the total number of students assigned to project topics that l_k considers as desirable as $t(p_j)$ is less than d_k. Therefore, if p is offered by l_k, then l_k prefers $t(p_j)$ to $t(p)$. Hence, $s \in M(l_k)$, l_k prefers $t(p_j)$ to $t(p)$, and (s, p_j) is a blocking pair in M. Now suppose that $s \notin M(l_k)$. Again, condition (3) implies that either l_k is undersubscribed in M, or there exists at least one student assigned to a project $t(p_z)$ where l_k prefers $t(p_j)$ to $t(p_z)$. If l_k is undersubscribed, then (s, p_j) is a blocking pair, since $s \notin M(l_k)$. Otherwise, l_k prefers $t(p_j)$ to their worst non-empty project topic. In this case, (s, p_j) still forms a blocking pair in M. Therefore, both conditions guarantee the existence of a blocking pair in M.

($\Leftarrow$) Conversely, suppose that there exists some blocking pair (s, p_j) in M, where s is a type-i student and p_j is a project offered by lecturer l_k. By definition of a blocking pair, the student s is either unassigned in M or assigned to some project p in M but prefers $t(p_j)$ to $t(p)$. Additionally, project p_j is undersubscribed in M, and at least one of the following conditions holds:

(a) $s \in M(l_k)$ and l_k prefers $t(p_j)$ to $t(p)$.

(b) $s \notin M(l_k)$ and l_k is undersubscribed in M.

(c) $s \notin M(l_k)$, but l_k prefers $t(p_j)$ to the topic of their worst non-empty project in M.

Since p_j is undersubscribed in M, the number of students assigned to p_j is strictly less than its capacity c_j, i.e., $\sum_{1 \leq i \leq r} X_{ij} < c_j$. Therefore condition (1) holds. Furthermore, since s is either unassigned or assigned to a project whose topic is less desirable than $t(p_j)$ from the perspective of type-i students, it follows that s is not assigned to any project whose topic type-i students consider at least as desirable as $t(p_j)$. If $s \notin M(l_k)$, then s is not assigned to any project whose topic l_k considers at least as desirable as $t(p_j)$. Alternatively, if $s \in M(l_k)$ then l_k prefers $t(p_j)$ to $t(p)$, then again s is not assigned to any project whose topic l_k considers at least as desirable as $t(p_j)$. Therefore, there exists at least one type-i student who is not assigned to any project whose topic is considered at least as desirable as $t(p_j)$ by either type-i students or by l_k. Hence, condition (2) holds.

In case (a), since s is assigned to a project topic worse than $t(p_j)$ from l_k's perspective, the total number of students assigned to project topics that l_k considers at least as desirable as $t(p_j)$ must be less than d_k, given that the total number of students assigned to l_k is at most d_k. Therefore, $\sum_{\substack{t(p_m) \succeq_k t(p_j) \\ 1 \leq i \leq r}} X_{im} < d_k$, and condition (3) holds. Now consider case (b), where l_k is undersubscribed in M. This implies that the total number of students assigned to l_k is less than d_k. Therefore, the total number of students assigned to projects whose topics l_k considers as desirable as $t(p_j)$ is also less than d_k. Hence, $\sum_{\substack{t(p_m) \succeq_k t(p_j) \\ 1 \leq i \leq r}} X_{im} < d_k$, and condition (3) holds. Finally, in case (c), since l_k prefers $t(p_j)$ to the topic of their worst non-empty project, there exists at least one student assigned to a project whose topic is worse than $t(p_j)$ from l_k's perspective. Again, since the total number of students assigned to l_k cannot exceed d_k, it follows that $\sum_{\substack{t(p_m) \succeq_k t(p_j) \\ 1 \leq i \leq r}} X_{im} < d_k$. Thus, all cases of a blocking pair lead to conditions (1), (2) and (3) of the lemma statement, completing the proof. □

3.4.4.1 Reducing to one project per topic for each lecturer

In this subsection, we show that given an instance I of SPA-PUC with k project topics, we can construct, in polynomial time, a corresponding instance I' in which each lecturer's preference

list contains exactly one project for each topic they offer. Moreover, the size of the largest stable matching in I' is equal to that in I.

Lemma 3.4.2. *Let I be an instance of* SPA-PUC *with k project topics. We can construct in polynomial-time an instance I' where each lecturer offers at most one project in every project topic and the size of the largest stable matching in I' is the same as the size of the largest stable matching in I.*

Construction of I': Let I be an instance of SPA-P with k project topics. We construct a new instance I' from I as follows: The sets of students and lecturers in I' are the same as those in I. The set of projects $\mathcal{P}'$ in I' is defined as follows. For each lecturer $l_k \in \mathcal{L}$, let T_k be the set of topics offered by l_k. For each $t \in T_k$, we introduce a single project p_t, and let $P'_k = \{p_t : t \in T_k\}$ be the set of projects offered by l_k in I'. Next, we define a mapping f such that for each project $p_j \in P_k$ with topic t, we set $f(p_j) = p_t$. That is, all projects with topic t in I are mapped to the same project p_t in I'.

In this way, the preference list of each lecturer in I' contains exactly one project per topic. Recall that in the original instance I, each project p_j offered by lecturer l_k has capacity equal to that of l_k. To maintain uniform capacities, we assign each new project $p_t \in P'_k$ the same capacity as the lecturer l_k who offers it. The capacity of each lecturer in I' remains the same as in I. In I, both students and lecturers are indifferent between projects within the same topic, but have strict preferences over different topics. Similarly in I', each student and lecturer inherits their strict preferences over new projects of different topics. We now prove the result in two parts. First, we show that the size of the largest stable matching in I' is at least that in I (Lemma 3.4.3). Then we show the reverse direction in Lemma 3.4.4. Together, these imply that Lemma 3.4.2 holds.

Lemma 3.4.3 (Forward direction)**.** *Let I be an instance of* SPA-PUC *with k project topics. Then we can construct, in polynomial time, an instance I' in which each lecturer offers at most one project per project topic, such that the size of the largest stable matching in I' is at least the size of the largest stable matching in I.*

Proof. Let M be a largest stable matching in I. We construct a matching M' in I' by assigning each student s_i to project $f(p_j)$ whenever $(s_i, p_j) \in M$. Clearly, every student assigned in M remains assigned in M'. Consider any project $p_t \in I'$ offered by l_k. By construction, each project $p_t \in P'_k$ corresponds to a subset of projects in I that belong to some topic t and are offered by lecturer l_k. Specifically, the set is defined as $f^{-1}(p_t) = \{p_j \in P_k \mid f(p_j) = p_t\}$. In I, each project $p_j \in f^{-1}(p_t)$ has capacity equal to the capacity of lecturer l_k. Furthermore, in I', the project p_t is assigned the same capacity as l_k, and the capacity of l_k remains unchanged between I and I'.

Since the total number of students assigned across all projects in P_k cannot exceed l_k's capacity, it follows that the total number of students assigned across all projects in $f^{-1}(p_t)$ in M cannot be more than l_k's capacity. Hence, the total number of students assigned to p_t in M' does not exceed

its capacity, and thus p_t is not oversubscribed in M'. Moreover, no lecturer is oversubscribed in M', since each lecturer l_k is assigned exactly the same set of students in M' as in M. Therefore, M' is a valid matching. We now prove that M' is stable.

Stability of M': Suppose that M' is not stable. Then there exists a blocking pair (s_i, p) in M', where p is offered by lecturer l_k, such that s_i is either unassigned in M', or strictly prefers p to $M'(s_i)$, p is undersubscribed in M', and one of the following conditions holds:

(a) $s_i \in M'(l_k)$, and l_k strictly prefers p to $M'(s_i)$, or

(b) $s_i \notin M'(l_k)$, and l_k is undersubscribed in M', or

(c) $s_i \notin M'(l_k)$, and l_k prefers $t(p)$ to their worst non-empty topic in M'.

Clearly, p and $M'(s_i)$ belong to different topics. By construction, each project p in I' corresponds to the set of projects in I that belong to topic t and are offered by lecturer l_k. Moreover, all students assigned to topic-t projects offered by l_k in M are assigned to p in M'. So if p is undersubscribed in M', then some topic-t project p_j in I must be undersubscribed in M. Now consider the student s_i. If s_i is unassigned in M', then they must also be unassigned in M, since all assigned students in M remain assigned in M'. Similarly, if s_i strictly prefers p to $M'(s_i)$, then they strictly prefer p_j to $M(s_i)$, since in M', no student is reassigned to a project on a different topic. We now show that in each of the three cases above, (s_i, p_j) blocks M, contradicting the stability of M.

Case (a): Since $s_i \in M'(l_k)$, it follows that $s_i \in M(l_k)$. Since $t(p_j) \neq t(M'(s_i))$, and students remain assigned to projects of the same topic in both M and M', we also have $t(p_j) \neq t(M(s_i))$. Moreover, since l_k strictly prefers p to $M'(s_i)$, and $M(s_i)$ belongs to the same topic as $M'(s_i)$, it follows that l_k also strictly prefers p_j to $M(s_i)$. Thus, s_i strictly prefers p_j to $M(s_i)$, p_j is undersubscribed in M, and l_k strictly prefers p_j to $M(s_i)$, so (s_i, p_j) forms a blocking pair in M, a contradiction.

Case (b): Since l_k is assigned the same set of students in M and M', it follows that l_k is also undersubscribed in M. Therefore, s_i strictly prefers p_j to $M(s_i)$, p_j is undersubscribed in M, and l_k is undersubscribed in M; thus, (s_i, p_j) blocks M, a contradiction.

Case (c): Since l_k strictly prefers $t(p)$ to their worst non-empty topic t_z in M', there exists a project $p_z \in M(l_k)$ belonging to topic t_z such that l_k strictly prefers p_j to p_z. Since s_i strictly prefers p_j to $M(s_i)$, p_j is undersubscribed in M, and l_k prefers $t(p_j)$ to $t(p_z)$, it follows that (s_i, p_j) blocks M, a contradiction.

Therefore, M' admits no blocking pairs and is stable. Since the same set of students are assigned in M' as in M, it follows that $|M'| = |M|$. □

Lemma 3.4.4 (Reverse direction)**.** *Let I' be the instance constructed from an instance I of* SPA-PUC*, where each lecturer offers at most one project per project topic. Then the size of the largest stable matching in I is at least the size of the largest stable matching in I'.*

Proof. Let M' be a largest stable matching in I'. We construct a matching M in I by assigning each student s_i to a single project $p_j \in \mathcal{P}$ such that $f(p_j) = p_t$ whenever $(s_i, p_t) \in M'$. This assignment is possible since each project p_j in $\mathcal{P}$ has capacity d_k, and the corresponding project p_t in $\mathcal{P}'$ also has capacity d_k. Clearly, each student is assigned to at most one project in M, since they are assigned to at most one project in M'. Since no more than d_k students are assigned to p_t in M', and each project p_j in $f^{-1}(p_t)$ has capacity d_k, it follows that we can assign all students from p_t in M' to a single project p_j in I without exceeding its capacity. Consequently, no project is oversubscribed in M. Finally, the set of students assigned to each lecturer in M is identical to that in M', so lecturer capacities are also respected. Hence, M is a valid matching in I. We now prove that M is stable.

Stability of M: Suppose that M is not stable. Then there exists a blocking pair (s_i, p_j) in M, where p_j is offered by lecturer l_k, such that s_i is either unassigned in M, or strictly prefers p_j to $M(s_i)$, p_j is undersubscribed in M, and one of the following conditions holds:

(a) $s_i \in M(l_k)$, and l_k strictly prefers p_j to $M(s_i)$, or

(b) $s_i \notin M(l_k)$, and l_k is undersubscribed in M, or

(c) $s_i \notin M(l_k)$, and l_k prefers $t(p_j)$ to their worst non-empty topic in M.

Clearly, $t(p_j) \neq t(M(s_i))$. Suppose p_j belongs to some topic t. By construction, the project $p_j \in I$ corresponds to some project $p_t \in I'$ such that $f(p_j) = p_t$. Similarly, let p_u denote the project in I' such that $f(M(s_i)) = p_u$. If s_i is unassigned in M, then s_i is also unassigned in M', since all assigned students in M remain assigned in M' to projects on the same topic. Similarly, if s_i strictly prefers p_j to $M(s_i)$, then s_i strictly prefers p_t to p_u, since in M', students are not reassigned to projects on different topics. We now show that in each of the three cases above, the pair (s_i, p_t) blocks M', contradicting the stability of M'.

Case (a): Since $t(p_j) \neq t(M(s_i))$, it follows that $p_t \neq p_u$. Moreover, since s_i is assigned in M to a project belonging to a different topic $t(M(s_i))$ offered by l_k, it follows that no other project topic offered by l_k can have d_k students assigned to it; otherwise, l_k would be oversubscribed in M. This implies that the total number of students assigned across the projects in $t(p_j)$ and offered by l_k is strictly less than d_k. Since p_t in I' has capacity d_k, it follows that p_t is undersubscribed in M'. Also, $s_i \in M(l_k)$ implies that $s_i \in M'(l_k)$. Moreover, since l_k strictly prefers p_j to $M(s_i)$, it follows that l_k prefers p_t to p_u. Thus, s_i strictly prefers p_t to p_u, p_t is undersubscribed in M', and l_k strictly prefers p_t to p_u, so (s_i, p_t) forms a blocking pair in M', a contradiction.

Case (b): Since l_k is assigned the same set of students in M and M', it follows that l_k is also undersubscribed in M'. Since the total number of students assigned to l_k in M is strictly less than d_k, it follows that the number of students assigned across the projects belonging to topic $t(p_j)$ and offered by l_k, is also strictly less than d_k. As p_t in I' contains these students and has capacity d_k, it follows that p_t is undersubscribed in M'. Therefore, s_i strictly prefers p_t to p_u, p_t is undersubscribed in M', and l_k is undersubscribed in M'; thus, (s_i, p_t) blocks M', a contradiction.

Case (c): Here, l_k prefers topic $t(p_j)$ to their worst non-empty topic t_z in M. Let $p \in I$ be a project in t_z assigned to some student in M. Then there exists some project $p_z \in P'_k$ such that $f(p) = p_z$, and l_k strictly prefers p_t to p_z. By a similar argument as in case (a), since there exists at least one student assigned to t_z, no other project topic offered by l_k can have d_k students assigned to it; otherwise, l_k would be oversubscribed in M. Consequently, the total number of students assigned to the projects in $t(p_j)$ offered by l_k is strictly less than d_k. Since p_t in I' has capacity d_k and contains these students, it follows that p_t is undersubscribed in M'. Moreover, since s_i strictly prefers p_t to p_u, and p_t is undersubscribed in M', it follows that (s_i, p_t) blocks M', a contradiction.

Therefore, M admits no blocking pairs and is stable. Since the same set of students are assigned in M' as in M, it follows that $|M| = |M'|$. □

3.4.4.2 Reducing to one lecturer per type

In this subsection, we show that given an instance I of SPA-PUC with t different lecturer types, we can construct, in polynomial time, a corresponding instance I' in which there is exactly one lecturer of each type. Furthermore, the size of the largest stable matching in I' is equal to that in I.

Lemma 3.4.5. *Given an instance I of SPA-PUC with k distinct lecturer types, we can construct, in polynomial time, a corresponding instance I' in which there is exactly one lecturer of each type. Furthermore, the size of the largest stable matching in I' is equal to that in I.*

By Lemma 3.4.2, we may assume without loss of generality that in a given instance I of SPA-PUC, each lecturer offers exactly one project for each topic. We adopt this assumption for the remainder of this section.

Construction of I': Let I be an instance of SPA-PUC with k project topics. We construct a new instance I' from I as follows. The set of students in I' is identical to that in I. For each type of lecturer t in I, let L_t denote the set of all lecturers of type t. In I', we create a single *combined* lecturer l_t to represent all lecturers in L_t. The capacity of l_t is set to $d_t = \sum_{l_k \in L_t} d_k$, where d_k is the capacity of each lecturer $l_k \in L_t$. For each topic q offered by lecturers in L_t, we create a single new project p_q offered by l_t. We define a mapping f such that for each project p_j belonging to topic q, we set $f(p_j) = p_q$. The capacity of p_q in I' is set to the capacity of all projects in P_k that belongs to topic q.

In I', l_t offers only these new projects p_q, one per topic, and both students and l_t inherit their strict preferences over project topics exactly as in I. Recall that each project p_j in I has capacity equal to the capacity of its lecturer l_k. Moreover, each lecturer in I offers at most one project on each topic. By construction of I', the capacity of each new project p_q is set to the total capacity of all original projects p_j under topic q, and offered by lecturers in L_t; this is also equal to the sum of the capacities of all lecturers in L_t. Since the combined lecturer l_t has capacity equal to the sum of the capacities of all lecturers in L_t, it follows that the capacity of each project p_q is exactly the same as capacity of l_t.

Transformation from M to M': Let M be the largest stable matching in I. We construct a matching M' in I' by assigning each student s_i to the project $f(p_j)$ whenever $(s_i, p_j) \in M$. Suppose p_j belongs to topic q and is offered by a type-t lecturer l_k. Let $f(p_j) = p_t$, where p_t denotes the project in I' that corresponds to the topic q projects offered by l_k in I. Furthermore, p_t is offered by the combined lecturer l_t in I'. Clearly, each student in I' is assigned to exactly one project in M'. Since the capacity of p_t is equal to the total capacity of all original projects under topic q, and none of these projects were oversubscribed in M, it follows that p_t is not oversubscribed in M'. Hence, no project is oversubscribed in M'.

Finally, each lecturer l_t in I' corresponds to the set of all type-t lecturers in I and is given a capacity equal to the combined capacities of those lecturers. The number of students assigned to l_t in M' is exactly the total number assigned to all type-t lecturers in M. Since no lecturer in M is oversubscribed, no lecturer is oversubscribed in M'. Hence, M' is a valid matching.

In Lemma 3.4.6, we show that the size of a largest stable matching M in I is the same as that of M', that is, $|M| = |M'|$.

Lemma 3.4.6. *Let M be a largest stable matching in I, and let M' be the matching in I' obtained via the construction described above. Then M' is stable in I', and $|M| = |M'|$.*

Proof. We note that by the construction of M', the total number of assigned students in M is the same in M', therefore $|M| = |M'|$. Now, suppose for contradiction that there exists a blocking pair (s_i, p_j) in M'. Let l_k be the lecturer who offers p_j. Then, s_i is either unassigned in M' or strictly prefers p_j to $M'(s_i)$, p_j is undersubscribed in M', and one of the following conditions hold:

(a) $s_i \in M'(l_k)$, and l_k strictly prefers p_j to $M'(s_i)$.

(b) $s_i \notin M'(l_k)$, and l_k is undersubscribed in M'.

(c) $s_i \notin M'(l_k)$, and l_k prefers $t(p_j)$ to their worst non-empty project topic in M'.

Suppose that l_k corresponds to the set of type-t lecturers in I. By construction, the project p_j in I' corresponds to the set of projects in I that belong to topic $t(p_j)$ and are offered by type-t lecturers.

Moreover, each student has the same preferences over project topics in I and I'. Consequently, if some student s_i is unassigned in M' or strictly prefers p_j to $M'(s_i)$, then s_i is unassigned in M or prefers all projects in $t(p_j)$ to $M(s_i)$. Now, we consider each blocking pair condition as follows:

Case (a): Here, $s_i \in M'(l_k)$ and l_k strictly prefers p_j to $M'(s_i)$. Let $M'(s_i) = p_z$. Since l_k offers both p_j and p_z in I', it follows that p_z corresponds to some project $\hat{p}$ in I, which is offered by a type-t lecturer l, such that $t(\hat{p}) = t(p_z)$ and $s_i \in M(\hat{p})$. Since l has the same preferences over topics as l_k, there exists some other project p offered by l such that $t(p) = t(p_j)$, and l strictly prefers p to $\hat{p}$. Similarly, s_i strictly prefers p to $\hat{p}$ since $t(p) = t(p_j)$. Since $s_i \in M(l)$ but is assigned to project $\hat{p}$, such that l strictly prefers p to $\hat{p}$, and each project offered by l has the same capacity as l, it follows that p is undersubscribed in M. Therefore, s_i strictly prefers p to $\hat{p}$, p is undersubscribed in M, $s_i \in M(l)$, and l strictly prefers p to $\hat{p}$. Thus, the pair (s_i, p) blocks M, a contradiction.

Case (b): Then $s_i \notin M'(l_k)$ and l_k is undersubscribed in M'. Since l_k is undersubscribed in M', the total number of students assigned to all type-t lecturers in M must be strictly less than their combined capacity. It follows that at least one type-t lecturer l is undersubscribed in M. Furthermore, since l_k offers p_j in I', there must exist some project p offered by l in I such that $t(p_j) = t(p)$. This follows from the construction of l_k, which ensures that l_k inherits the same preferences over topics as each type-t lecturer. Since l is undersubscribed in M, p must also be undersubscribed in M, given that both p and l have the same capacities. Moreover, since $s_i \notin M'(l_k)$, it follows that s_i is not assigned to any type-t lecturer in M, because by the construction of M', every student assigned to a lecturer of type t in M must be assigned to l_k in M'. Therefore, $s_i \notin M(l)$; moreover, both p and l are undersubscribed in M. Since s_i is assigned in M and prefers all projects in $t(p_j)$ to $M(s_i)$, it follows that s_i strictly prefers p to $M(s_i)$, given that $t(p) = t(p_j)$. In this case, the pair (s_i, p) is a blocking pair in M, a contradiction.

Case (c): Then $s_i \notin M'(l_k)$ and l_k prefers $t(p_j)$ to the worst non-empty project topic in $M'(l_k)$. Let p_z be some non-empty project in $M'(l_k)$ where $t(p_z)$ is the worst project topic for l_k in M'. Then, l_k prefers $t(p_j)$ to $t(p_z)$. Let $\hat{p}$ be a corresponding non-empty project in I such that $f(\hat{p}) = p_z$ and $t(\hat{p}) = t(p_z)$; let l be the lecturer in I who offers $\hat{p}$. Clearly, l has the same preferences over topics as l_k does. Therefore, l offers some project p such that $t(p) = t(p_j)$ and l prefers $t(p)$ to $t(\hat{p})$. Since $s_i \notin M'(l_k)$, it follows that s_i is not assigned to any type-t lecturer in M, and therefore, $s_i \notin M(l)$. Moreover, since s_i prefers all projects in $t(p_j)$ to $t(M'(s_i))$, it follows that s_i strictly prefers p to $M(s_i)$. Since $\hat{p}$ is non-empty, then no other project offered by l in M can be full (Otherwise l is oversubscribed in M). Hence, p is undersubscribed in M. In this case, s_i is either unassigned in M or strictly prefers p to $\hat{p}$, p is undersubscribed in M, $s_i \notin M(l)$, and l prefers $t(p)$ to $t(\hat{p})$. Thus, the pair (s_i, p) blocks M, a contradiction.

Hence, if M is the largest stable matching in I, then the corresponding matching M' is stable and $|M| = |M'|$. This concludes the proof. □

We now prove the reverse direction: if M' is a largest stable matching in I', then the corresponding matching M is stable and satisfies $|M| = |M'|$. To establish this, we first define the notion of *unavailable topics* in I', which plays a key role in the proofs of the subsequent lemmas. The proof involves transforming M' into an intermediate matching M_0 in I', and showing that $|M'| = |M_0|$. In M_0, each student is either assigned the same project as in M', or is assigned to a more preferred project in M_0 offered by the same lecturer, provided the project belongs to an available topic. We then construct a matching M in the original instance I from M_0, prove that M is stable, and show that $|M| = |M_0|$.

Definition 3.4.2 (Unavailable Topics)**.** For each lecturer $l \in I'$, a topic t' is said to be unavailable for l if there exists a student s, who is either unassigned in M' or assigned to a project offered by a different lecturer, and a topic $\hat{t}$ offered by l, such that:

- s prefers topic $\hat{t}$ to the topic of their assigned project in M', and
- l prefers topic $\hat{t}$ to topic t'.

A topic is *available* for l if it is not unavailable.

Transformation from M' to M_0: Let M' be a largest stable matching in I'. We construct a matching M_0 in I' by initially assigning each student to the same project and lecturer as in M'. We define a *swap* as follows: for a student s_i, let p be a project on s_i's preference list such that: (i) s_i strictly prefers p to $M_0(s_i)$; (ii) p is undersubscribed in M_0; (iii) both p and $M_0(s_i)$ are offered by the same lecturer l; and (iv) the topic $t(p)$ is an *available topic* for l. We say that a *feasible swap* exists in M_0 whenever these conditions are satisfied. If such a project p exists, we remove the assignment between s_i and $M_0(s_i)$, and assign s_i to p. We greedily apply swaps until no feasible swap remains in M_0, at which point the construction terminates with the matching M_0.

Observation 3.4.1. *If a topic is available before a feasible swap in M', then it remains available in M_0.*

Lemma 3.4.7. *Let M' be a stable matching in I', and let M_0 be the matching obtained from M' via the construction described above. Then M_0 is stable and $|M_0| = |M'|$.*

Proof. It is straightforward to verify that M_0 is a valid matching. We observe that each student either remains assigned to the same project as in M' or is moved to a more preferred project, hence no student is multiply assigned. Since swaps only occur between projects offered by the same lecturer, each lecturer has the same set of students in M_0 and M'. Therefore, no lecturer is oversubscribed in M_0. Furthermore, a student is only moved to a project if that project is undersubscribed, so no project is oversubscribed in M_0. Suppose, for contradiction, that there exists a blocking pair (s_i, p_j) in M_0. Then s_i is either unassigned in M_0 or strictly prefers p_j to $M_0(s_i)$, p_j is undersubscribed in M_0 (where p_j is offered by l_k), and one of the following conditions holds:

(a) $s_i \in M_0(l_k)$, and l_k strictly prefers p_j to $M_0(s_i)$,

(b) $s_i \notin M_0(l_k)$, and l_k is undersubscribed in M_0,

(c) $s_i \notin M_0(l_k)$, and l_k prefers $t(p_j)$ to their worst non-empty project topic in M_0.

Case (a): Since $s_i \in M_0(l_k)$ and students remain assigned to the same lecturer in the construction of M_0, it follows that $s_i \in M'(l_k)$. If s_i strictly prefers p_j to $M_0(s_i)$, then the same preference holds with respect to $M'(s_i)$, since students are only moved to more preferred projects during the construction of M_0. Thus, s_i strictly prefers p_j to $M'(s_i)$. Let $p_a = M_0(s_i)$ and $p_b = M'(s_i)$. Clearly, $t(p_j) \neq t(p_a)$. We now claim that $t(p_j)$ is an available topic for l_k in M'. Suppose, for contradiction, that it is not available. Then there exists a student $s \notin M'(l_k)$ and a topic $\hat{t}$ offered by l_k such that s prefers $\hat{t}$ to their assigned topic in M', and l_k prefers $\hat{t}$ to $t(p_j)$. We consider two subcases depending on whether or not s_i was involved in a swap.

Case (i): $p_a = p_b$. Then s_i was not involved in a swap. In this case, l_k prefers $t(p_j)$ to $t(p_b)$. Since s_i is assigned in M' to a different project p_b offered by l_k, no other project offered by l_k is full in M'. Thus, each project in $\hat{t}$ is undersubscribed in M'. Consequently, s prefers some project p in $\hat{t}$, p is undersubscribed in M', and l_k prefers $\hat{t}$ to the non-empty project topic $t(p_b)$. Hence, (s, p) blocks M', a contradiction.

Case (ii): $p_a \neq p_b$. Then s_i was involved in a swap; moreover, s_i strictly prefers p_j to p_a, and p_a to p_b. This holds because each student is assigned to a more preferred project in M_0. Also, since s_i is assigned to p_a in M_0, the topic $t(p_a)$ is available for l_k in M'. If l_k prefers $t(p_j)$ to $t(p_b)$, then, following the same argument as in case (i), we arrive at a contradiction involving the pair (s, p). Now suppose l_k prefers $t(p_j)$ to $t(p_a)$. Since l_k prefers $\hat{t}$ to $t(p_j)$, and prefers $t(p_j)$ to $t(p_a)$, it follows that l_k prefers $\hat{t}$ to $t(p_a)$. By our assumption, there exists a student $s \notin M'(l_k)$ who prefers $\hat{t}$ to their assigned topic in M'. However, by definition, this implies that $t(p_a)$ is unavailable. This contradicts the fact that $t(p_a)$ must have been available in M' for s_i to be assigned to p_a in M_0.

Thus, in both cases (i) and (ii), $t(p_j)$ is an available topic for l_k in M', and consequently also in M_0. Furthermore, as p_j is undersubscribed in M_0, the pair (s_i, p_j) satisfies the conditions for a feasible swap. Thus, s_i should have been assigned to p_j during the construction of M_0, contradicting the fact that M_0 was obtained by terminating only when no such pair remains. Therefore, there are no blocking pairs of case (a) in M_0.

Case (b): Then $s_i \notin M_0(l_k)$ and l_k is undersubscribed in M_0. Since students are only moved between projects offered by the same lecturer, it follows that $s_i \notin M'(l_k)$ as well. Moreover, if s_i is unassigned in M_0 or prefers $t(p_j)$ to $t(M_0(s_i))$, then the same holds in M'; that is, s_i is either unassigned in M' or prefers $t(p_j)$ to $t(M'(s_i))$. Since the set of students assigned to l_k is the same in M_0 and M', it follows that if l_k is undersubscribed in M_0, then l_k is also undersubscribed in M'. If p_j is undersubscribed in M', then the pair (s_i, p_j) blocks M', contradicting the stability of

M'. Now suppose that p_j is undersubscribed in M_0 but full in M'. Then, during the construction of M_0, some student $s \in M'(l_k)$ must have been moved from p_j to another project p offered by l_k.

By the construction of M_0, s strictly prefers $t(p)$ to $t(p_j)$, project p is undersubscribed in M_0, and $t(p)$ is an available topic for l_k. Since $t(p)$ is available, by definition, there is no student $s' \notin M'(l_k)$ who prefers some topic $\hat{t}$ to their assignment in M' and l_k prefers $\hat{t}$ to $t(p)$. However, $s_i \notin M'(l_k)$, and since s_i prefers $t(p_j)$ to their assignment in M', it follows that l_k does not prefer $t(p_j)$ to $t(p)$; otherwise, s_i would be a student violating the fact that $t(p)$ is available, a contradiction. Therefore, l_k prefers $t(p)$ to $t(p_j)$. But this means that s strictly prefers p to p_j, p is undersubscribed in M', and l_k is also undersubscribed in M', so the pair (s, p) blocks M', a contradiction. Therefore, there are no case (b) blocking pairs in M_0.

Case (c): Suppose $s_i \notin M_0(l_k)$, and l_k prefers $t(p_j)$ to their worst non-empty project topic in M_0. Since students are only moved between projects offered by the same lecturer, it follows that $s_i \notin M'(l_k)$ as well. Moreover, if s_i is unassigned in M_0 or prefers $t(p_j)$ to $t(M_0(s_i))$, then the same must hold in M'; that is, s_i is either unassigned in M', or prefers $t(p_j)$ to $t(M'(s_i))$. Let p_z be a non-empty project offered by l_k such that $t(p_z)$ is the worst topic for l_k in M_0. Since l_k's preferences over topics remain the same in M_0 and M', it follows that l_k also prefers $t(p_j)$ to $t(p_z)$ in M'. If p_j is undersubscribed in M', then (s_i, p_j) forms a blocking pair in M', since $s_i \notin M'(l_k)$, s_i strictly prefers p_j to $M'(s_i)$, and l_k prefers $t(p_j)$ to the non-empty $t(p_z)$. Thus, the only way (s_i, p_j) does not block M' is if either (i) p_z is non-empty in M_0 but empty in M', or (ii) p_j is undersubscribed in M_0 but full in M'. We consider each of these cases below.

Case (i): Suppose that p_z is empty in M' but is non-empty in M_0. Then there exists a student $s_z \in M'(l_k)$ who was not assigned to p_z in M' but assigned to it in M_0; that is, $s_z \in M_0(p_z) \setminus M'(p_z)$. This implies p_z was undersubscribed in M', and $t(p_z)$ was an available topic for l_k. By the definition of available topics, this means there is no student $s \notin M'(l_k)$ who prefers some topic $\hat{t}$ to their assignment in M', and l_k prefers $\hat{t}$ to $t(p_z)$. However, since $s_i \notin M'(l_k)$, and s_i prefers $t(p_j)$ to their assignment in M', and l_k prefers $t(p_j)$ to $t(p_z)$, this would imply that $t(p_z)$ is not available, a contradiction. Therefore, $t(p_z)$ must be non-empty in M'.

Case (ii): Now suppose p_j is full in M' but undersubscribed in M_0. Then, during the construction of M_0, some student $s \in M'(l_k)$ must have moved from p_j to another project p offered by l_k, i.e., $M'(s) = p_j$. By the definition of feasible swaps, s strictly prefers p to p_j, p is undersubscribed in M_0, and $t(p)$ is an available topic for l_k. Since $t(p)$ is available, there cannot exist a student $s' \notin M'(l_k)$ who prefers some topic $\hat{t}$ to their assignment in M' such that l_k prefers $\hat{t}$ to $t(p)$. But $s_i \notin M'(l_k)$, and s_i prefers $t(p_j)$ to $t(M'(s_i))$, so if l_k preferred $t(p_j)$ to $t(p)$, then $t(p)$ would not be available — a contradiction. Hence, l_k prefers $t(p)$ to $t(p_j)$. It follows that (s, p) blocks M', since $s \in M'(l_k)$, s strictly prefers p to p_j, p is undersubscribed in M', and l_k prefers $t(p)$ to $t(p_j)$. This contradicts the stability of M'. We conclude that no blocking pair of type (c) exists in M_0.

Hence, M_0 is a stable matching. Moreover, since each student assigned in M' remains assigned in M_0, it follows that $|M_0| = |M'|$. This concludes the proof. □

Transformation from M_0 to M: We construct a matching M in the original instance I by using the assignments in M_0. Specifically, if a student s is assigned to a project p in M_0, where p is offered by the combined lecturer l_t in I' (corresponding to type-t lecturers in I), we assign s to some project p_j in I such that $f(p_j) = p$ and p_j is offered by a lecturer of type t. We distribute the students assigned to p across each project p_j in I where $f(p_j) = p$, and p_j is offered by a lecturer of type t. This distribution is carried out so that the capacity of each individual project p_j in I is not exceeded.

Since the total capacity of the combined project p in I' is equal to the sum of the capacities of all projects in $f^{-1}(p)$, there is enough space to distribute all students without any project being oversubscribed. Furthermore, no lecturer is oversubscribed in M, since each l_t in I' has a capacity equal to the total capacity of the type-t lecturers in I, and each project in I has the same capacity as the lecturer who offers it. Moreover, each student is assigned to exactly one project in M. Therefore, M is a valid matching.

Lemma 3.4.8. *Let M be a matching in I obtained via the construction described above. Then M admits no blocking pair (s_i, p_j) where p_j and $M(s_i)$ are offered by lecturers of different types.*

Proof. Suppose, for contradiction, that M admits a blocking pair (s_i, p_j) whereby s_i is either unassigned in M or s_i strictly prefers p_j to $M(s_i)$, and projects p_j and $M(s_i)$ are offered by lecturers of different types. By construction, each student s has the same preferences over projects and project topics in M and M_0. Therefore, s_i is either unassigned in M_0 or s_i strictly prefers p_j to $M_0(s_i)$. Let l_k denote the lecturer who offers p_j in I. Clearly, $s_i \notin M(l_k)$ since both p_j and $M_0(s_i)$ are offered by different types of lecturers. Since (s_i, p_j) blocks M, then p_j is undersubscribed in M, and one of the following holds:

(a) $s_i \notin M(l_k)$ and l_k is undersubscribed in M, or

(b) $s_i \notin M(l_k)$ and l_k prefers $t(p_j)$ to their worst non-empty project topic in M.

Suppose that l_k is a type-t lecturer. By construction of M, all type-t lecturers in I correspond to a single lecturer l_t in I', and l_t offers a combined project p such that $f(p_j) = p$ and $t(p_j) = t(p)$. We recall that, by assumption, each lecturer in I offers at most one project on each topic. Since p_j is undersubscribed in M, the total number of students assigned across all projects in $t(p_j)$ and offered by lecturers of type-t in I is less than their combined capacity. It follows that the corresponding project p is also undersubscribed in M_0. Moreover, since p_j and $M(s_i)$ are offered by lecturers of different types, it follows that $s_i \notin M_0(l_t)$. We consider the possible blocking pair cases as follows:

Case (a): Suppose both p_j and l_k are undersubscribed in M. Since l_k is undersubscribed in M, it follows that the total number of students assigned to type-t lecturers in M is strictly less than their total capacity. Hence, l_t is undersubscribed in M_0. Moreover, p is undersubscribed in M_0. Therefore, $s_i \notin M_0(l_t)$, s_i is either unassigned in M_0 or s_i strictly prefers p to $M_0(s_i)$, both p and l_t are undersubscribed in M_0. Thus, (s_i, p) forms a blocking pair in M_0, a contradiction.

Case (b): Suppose p_j is undersubscribed in M, and l_k prefers $t(p_j)$ to the topic of their worst non-empty project in M_0, say $t(p_z)$. By construction, $t(p_z)$ corresponds to some project $\hat{p}$ offered by l_t, where $t(p_z) = t(\hat{p})$. Moreover, since $t(p) = t(p_j)$, l_t also prefers $t(p)$ to $t(\hat{p})$. Also, p is undersubscribed in M_0. In this case, $s_i \notin M_0(l_t)$, either s_i is unassigned in M_0 or s_i strictly prefers p to $M_0(s_i)$, p undersubscribed in M_0, and l_t prefers $t(p)$ to $t(p_z)$. Therefore, (s_i, p) forms a blocking pair in M_0, contradicting Lemma 3.4.7.

Since both cases (a) and (b) lead to a contradiction, we conclude that M admits no such blocking pairs. □

Lemma 3.4.9. *Let M_0 be a stable matching in I' with no feasible swaps, and let M be the matching in I obtained from M_0 via the construction described above. Then M is stable and $|M| = |M_0|$.*

Proof. Suppose for contradiction that M is not stable. Then there exists a blocking pair (s_i, p_j) in M. By Lemma 3.4.8, the blocking pair (s_i, p_j) involves lecturers of the same type, that is, p_j and $M(s_i)$ are offered by lecturers of the same type. Let l_k be the lecturer who offers p_j. It follows that s_i is either unassigned in M or strictly prefers p_j to $M(s_i)$, p_j is undersubscribed in M, and one of the following holds:

(a) $s_i \in M(l_k)$, and l_k strictly prefers p_j to $M(s_i)$,

(b) $s_i \notin M(l_k)$, and l_k is undersubscribed in M,

(c) $s_i \notin M(l_k)$, and l_k prefers $t(p_j)$ to their worst non-empty topic in M.

Suppose that l_k is a type-t lecturer. Let l_t be the lecturer in I' corresponding to the type-t lecturers in I. Then l_t offers a project p in I' such that $f(p_j) = p$. Similarly, let $f(M(s_i)) = \hat{p}$. We note that if s_i is unassigned in M, then by construction, s_i is also unassigned in M_0, since every student assigned in M is also assigned in M_0. Similarly, if s_i strictly prefers p_j to $M(s_i)$, then s_i also strictly prefers p to $\hat{p}$, since $t(p) = t(p_j)$ and $t(\hat{p}) = t(M(s_i))$; also, each student's preferences over project topics are identical in M and M_0. Since p_j is undersubscribed in M, it follows that the total number of students assigned across all projects with $f(p_j) = p$ and offered by type-t lecturers in I is less than their combined capacity. Therefore, p is undersubscribed in M_0. We now consider each case separately:

Case (a): Since $s_i \in M(l_k)$, it follows by construction that $s_i \in M_0(l_t)$. Furthermore, since l_k strictly prefers p_j to $M(s_i)$, it follows that l_t strictly prefers p to $\hat{p}$. Therefore, s_i is either

unassigned in M_0, or s_i strictly prefers p to $\hat{p}$, p is undersubscribed in M_0, and l_t strictly prefers p to $\hat{p}$. It follows that (s_i, p) blocks M_0, contradicting the stability of M_0.

Cases (b) and (c): Since $s_i \notin M(l_k)$, but both p_j and $M(s_i)$ are offered by lecturers of the same type (type t), it follows by construction that $s_i \in M_0(l_t)$. In case (b), if l_k is undersubscribed in M, then the total number of students assigned across all type-t lecturers is strictly less than their combined capacity. This implies that l_t is undersubscribed in M_0. In case (c), let p' be some non-empty project in the worst topic in $M(l_k)$, and let $f(p') = \hat{p}$. Then l_t prefers p to $\hat{p}$.

We now show that $t(p)$ is an available topic for l_t in M_0. Suppose not. Then, by the definition of unavailable topics, there exists some student $s \notin M_0(l_t)$ and some topic t^* offered by l_t such that: (i) s is either unassigned in M_0 or prefers t^* to the topic of their current project in M_0; and (ii) l_t prefers t^* to $t(p)$. By construction, each project offered by l_t has capacity equal to that of l_t. Since $s_i \in M_0(l_t)$, it follows that no project (other than $M_0(s_i)$) offered by l_t is full in M_0, and thus there exists a project p^* in topic t^* that is undersubscribed in M_0. In case (b), it follows that $s \notin M_0(l_t)$, s prefers $t(p^*)$ to $t(M_0(s))$, and both p^* and l_t are undersubscribed in M_0; thus, the pair (s, p^*) blocks M_0. In case (c), it follows that p^* is undersubscribed in M_0, and l_t prefers $t(p^*)$ to $t(\hat{p})$. Therefore, the pair (s, p^*) blocks M_0, contradicting its stability. Hence, $t(p)$ is an available topic for l_t in M_0.

Hence, $s_i \in M_0(l_t)$, s_i strictly prefers p to $M_0(s_i)$, p is undersubscribed in M_0, both p and $\hat{p}$ are offered by l_t, and $t(p)$ is an available topic for l_t. Thus, all conditions for a feasible swap between s_i and p are satisfied in M_0, contradicting the assumption that M_0 admits no feasible swaps. Therefore, M is a stable matching. Since the same set of students assigned in M_0 are assigned in M by construction, it follows that $|M| = |M_0|$, completing the proof. □

Since $|M'| = |M_0|$ and $|M_0| = |M|$, it follows that $|M'| = |M|$. Therefore, if M' is the largest stable matching in I', then we can construct a stable matching M in I such that $|M'| = |M|$. Together, Lemmas 3.4.6 – 3.4.9 prove that the transformation from I to I' preserves the size of the largest stable matching in both directions. Since this transformation can be carried out in polynomial time, Lemma 3.4.5 holds.

3.4.5 An ILP for SPA-PUC

An instance of INTEGER LINEAR PROGRAMMING model (ILP) consists of an integer matrix $A \in \mathbb{Z}^{m \times k}$, a vector $b \in \mathbb{Z}^m$, and an objective vector $c \in \mathbb{Z}^k$. The goal is to find a vector $x \in \mathbb{Z}^k$ that minimises the objective $c^\top x$, subject to the constraint $Ax \leq b$, or to determine that no feasible solution exists. While solving an ILP is NP-hard in general, a celebrated result by Lenstra [31] shows that the problem is fixed-parameter tractable (FPT) when parameterised by the number of variables (See Theorem 3.4.1). Specifically, an ILP with k variables can be solved in time $f(k) \cdot \text{poly}(W)$, where $f(k)$ is an exponential function depending only on k, and W is the total

size of the input. We use this result to show that the problem of finding a largest stable matching in SPA-PUC is FPT when parameterised by the number of project topics.

Theorem 3.4.1 ([31], based on [42, 76, 93]). *An Integer Linear Programming instance of size W with p variables can be solved using*

$$O\left(p^{2.5p+o(p)} \cdot (W + \log M_x) \log(M_x M_c)\right)$$

arithmetic operations and space polynomial in $W + \log M_x$, where M_x is an upper bound on the absolute value a variable can take in a solution, and M_c is the largest absolute value of a coefficient in the vector c.

We construct an ILP that maximises the total number of students across all types who are assigned to each project. Let I be an instance of SPA-PUC involving a set $S = \{s_1, s_2, \ldots, s_{n_1}\}$ of students, a set $P = \{p_1, p_2, \ldots, p_{n_2}\}$ of projects, and a set $L = \{l_1, l_2, \ldots, l_{n_3}\}$ of lecturers. Each student belongs to a type in the set $\{1, \ldots, r\}$, each lecturer belongs to a type in the set $\{1, \ldots, t\}$, and each project p_j belongs to a corresponding project topic. Suppose that p_j is offered by lecturer l_k, and let P_k be the set of projects offered by l_k. For each student type $i \in [r]$ and project $p_j \in P$, we introduce an integer variable $x_{i,j} \in \mathbb{Z}_{\geq 0}$ representing the number of students of type i assigned to project p_j.

By Lemma 3.4.1, a type-i student and project p_j can form a blocking pair only if *all three* of the lemma conditions hold simultaneously. To ensure that there are no blocking pairs, it therefore suffices to guarantee that for every such pair, *at least one* of these conditions fails. To model this in the ILP, we introduce binary variables $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j} \in \mathbb{Z} \quad \forall i, j$ for each student type i and project p_j pair. Each variable is used to ensure that the corresponding condition in Lemma 3.4.1 does not hold. Recall that $n_1 = \sum_{i=1}^{r} N_i$ is the total number of students in the instance. We use n_1^3 as a sufficiently large value in the blocking pair constraints so that when a binary variable (e.g., $\alpha_{i,j}$) is set to 1, the corresponding inequality becomes trivially satisfied. When the binary variable is 0, the constraint forces the corresponding blocking pair condition to fail.

Using the ILP variables $x_{i,j}$, we restate the three conditions from Lemma 3.4.1 under which a type-i student and project p_j form a blocking pair.

(a) $\sum_{i=1}^{r} x_{i,j} < c_j$,

(b) $N_i - \left(\sum_{t(p_m) \succeq_i t(p_j)} x_{i,m} + \sum_{t(p_m) \succeq_k t(p_j)} x_{i,m} - \sum_{\substack{t(p_m) \succeq_i t(p_j) \\ t(p_m) \succeq_k t(p_j)}} x_{i,m} \right) > 0$,

(c) $\sum\limits_{\substack{t(p_m)\succeq_k t(p_j)\\1\le i\le r}} x_{i,m} < d_k$,

We now present the ILP formulation:

Objective:

$$\text{Maximise} \quad \sum_{i=1}^{r}\sum_{j=1}^{n_2} x_{i,j}$$

That is, we aim to maximise the total number of students across all types who are assigned to all projects. Let A_i be the set of projects acceptable to each student type i.

Matching constraints:

$$\sum_{j\in A_i} x_{i,j} \le N_i \qquad \forall i \in [r] \quad \text{(no type-}i\text{ student is assigned to multiple projects)} \tag{1}$$

$$\sum_{i=1}^{r} x_{i,j} \le c_j \qquad \forall j \in [n_2] \quad \text{(no project is oversubscribed)} \tag{2}$$

$$\sum_{p_j\in P_k}\sum_{i=1}^{r} x_{i,j} \le d_k \qquad \forall k \in [n_3] \quad \text{(no lecturer is oversubscribed)} \tag{3}$$

Blocking pair constraints (for all $1 \le i \le r$ and $1 \le j \le n_2$):

$$\sum_{i=1}^{r}\sum_{j=1}^{n_2} x_{i,j} - c_j + n_1^3\alpha_{i,j} > 0 \tag{4}$$

$$N_i - \left(\sum_{t(p_m)\succeq_i t(p_j)} x_{i,m} + \sum_{t(p_m)\succeq_k t(p_j)} x_{i,m} - \sum_{\substack{t(p_m)\succeq_i t(p_j)\\ t(p_m)\succeq_k t(p_j)}} x_{i,m} \right) - n_1^3\beta_{i,j} < 0 \tag{5}$$

$$\sum_{\substack{t(p_m)\succeq_k t(p_j)\\1\le i\le r}} x_{i,m} - d_k + n_1^3\gamma_{i,j} > 0 \tag{6}$$

$$\alpha_{i,j} + \beta_{i,j} + \gamma_{i,j} \le 2 \tag{7}$$

In constraints (4)–(6), we use the binary variables to determine whether the corresponding blocking pair condition in Lemma 3.4.1 is enforced. Setting a variable to 0 ensures the corresponding condition fails; setting it to 1 causes the constraint to be trivially satisfied via the large value n_1^3. Explicitly:

- If $\alpha_{i,j} = 0$, then constraint (4) reduces to $\sum\limits_{i=1}^{r}\sum\limits_{i=1}^{n_2} x_{i,j} > c_j$, so condition (a) fails.
- If $\beta_{i,j} = 0$, then constraint (5) becomes negative, so condition (b) fails.
- If $\gamma_{i,j} = 0$, then constraint (6) reduces to $\sum x_{i,m} > d_k$, so condition (c) fails.

Constraint (7) ensures that for each type-i student and project p_j pair, at least one condition fails. Hence, no pair satisfies all blocking pair conditions simultaneously.

Lemma 3.4.10. *Let I be an instance of* SPA-PUC*. If the ILP described above admits a feasible solution S, then S corresponds to a stable matching M in I, where $obj(S) = |M|$.*

Proof. Suppose that the ILP above admits a feasible solution S. For each student type $i \in [r]$ and each project $p_j \in [n_2]$, the variable $x_{i,j}$ gives us the number of type-i students assigned to project p_j. From this, we construct a matching M in the original instance I as follows. Let N_i denote the number of students of type i. For each project p_j, we assign exactly $x_{i,j}$ different students to p_j, ensuring that no student is assigned to more than one project. This construction is valid since constraint (1) guarantees that the total number of assigned type-i students does not exceed N_i. Let M be the matching defined by the solution S. The total number of students assigned in M is

$$|M| = \text{obj}(S) = \sum_{i=1}^{r} \sum_{j=1}^{n_2} x_{i,j}.$$

Constraints (1)–(3) ensure that M is a valid matching: no student is assigned to more than one project, and the capacity constraints for projects and lecturers are respected. Suppose for contradiction that M contains a blocking pair involving some student s and project p_j; suppose s is a type-i student. Then, all three conditions in the statement of Lemma 3.4.1 must be satisfied for this pair. In the ILP, these blocking pair conditions are captured by constraints (4)–(6), and each is associated with a binary variable: $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$. These variables are designed so that setting a variable to 0 ensures its corresponding blocking pair condition *fails*. Therefore, if all three conditions hold for M, then any feasible assignment to $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ must set each variable to 1. But this would violate constraint (7), which requires

$$\alpha_{i,j} + \beta_{i,j} + \gamma_{i,j} \leq 2.$$

Hence, such a blocking pair cannot exist in M, and we conclude that M is stable. □

Lemma 3.4.11. *Let I be an instance of* SPA-PUC*, and let M be a stable matching in I. Then there exists a feasible solution to the ILP defined above with objective value $|M|$.*

Proof. Let M be a stable matching in instance I. For each student type $i \in [r]$ and project $p_j \in P$, define $x_{i,j}$ to be the number of type-i students assigned to project p_j in M. Since M is a valid matching, it satisfies the following: No student is assigned to more than one project, so for each $i \in [r]$, we have $\sum\limits_{j \in A_i} x_{i,j} \leq N_i$, satisfying constraint (1). No project is oversubscribed, so for each $j \in [n_2]$, we have $\sum_{i=1}^{r} x_{i,j} \leq c_j$, satisfying constraint (2). No lecturer is oversubscribed, so for each $k \in [n_3]$, we have $\sum_{p_j \in P_k} \sum_{i=1}^{r} x_{i,j} \leq d_k$, satisfying constraint (3).

We now show that the blocking pair constraints (4)–(7) can be satisfied by assigning suitable values to the binary variables $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j} \in \{0, 1\}$ based on the stable matching M. Since M is stable, it follows that for every student type $i \in [r]$ and every project $p_j \in \mathcal{P}$, at least one of the blocking pair conditions in Lemma 3.4.1 does not hold. For each such combination of student type and project, we identify a condition that fails in M, and set the corresponding binary variable, either $\alpha_{i,j}$, $\beta_{i,j}$, or $\gamma_{i,j}$, to 0. This ensures that the relevant constraint among (4), (5), or (6) is satisfied.

In this way, we construct a valid setting of the binary variables that satisfies the blocking pair constraints for that student type and project. Moreover, since at least one condition fails, we assign the value 1 to at most two of the variables, ensuring that constraint (7), which requires $\alpha_{i,j} + \beta_{i,j} + \gamma_{i,j} \leq 2$, is also satisfied. Thus, for every combination of student type i and project p_j, we can construct a valid assignment to the variables $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ such that constraints (4)–(7) are satisfied.

Therefore, the assignment derived from the stable matching M satisfies all constraints and defines a feasible solution to the ILP. The corresponding value of the objective function is

$$\text{obj}(S) = \sum_{i=1}^{r} \sum_{j=1}^{n_2} x_{i,j} = |M|,$$

as required. □

Theorem 3.4.2. *Let I be an instance of SPA-PUC with k project topics. Then finding a largest stable matching in I is fixed-parameter tractable when parameterised by k.*

Proof. We begin by applying the reduction of Lemma 3.4.2, which transforms the input instance I into an instance I_1 in which each lecturer offers at most one project per topic. We then apply Lemma 3.4.5 to obtain an instance I_2 in which there is only one lecturer per type. These reductions preserve the size of the largest stable matching and can be computed in polynomial time.

We then formulate the transformed instance I_2 as an ILP, as described in Section 3.4.5. This ILP is expressed in terms of student types, project topics, and lecturer types. In the worst case, each student type is characterised by a strict preference list over the k project topics, where students are indifferent between all projects belonging to the same topic. There are at most $k! < k^k$ such strict preference lists. Since students may only find a subset of topics acceptable, each list can be truncated in at most k ways. Hence, the total number of distinct student types is bounded by $k \cdot k^k = k^{k+1}$. On the lecturer side, their total number is bounded by $2^k - 1$.

In I_2, each lecturer offers at most one project per topic, so each lecturer offers at most k projects. Since there are at most $2^k - 1$ lecturer types, and each may offer up to k projects, the total number

of projects is at most

$$k(2^k - 1) = k2^k - k.$$

Given that there are k^{k+1} student types, the total number of student-type/project pairs is at most

$$k^{k+1}\Big(k2^k - k\Big) = k^{k+2}(2^k - 1).$$

The ILP includes:

- One variable $x_{i,j}$ for each student type i and project p_j;
- Three binary variables $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}$ per such pair;
- Three matching constraints: one per student type, one per project, and one per lecturer;
- Four blocking pair constraints for each student type and project pair.

Hence, the ILP has

$$O\Big(k^{k+2}(2^k - 1)\Big)$$

variables and constraints. Each variable $x_{i,j}$ is bounded above by n_1, the number of students, so the input size is

$$O\Big(k^{k+2}(2^k - 1)\log n_1\Big).$$

By Theorem 3.4.1, the ILP can be solved in time

$$O\left(\Big(k^{k+2}(2^k - 1)\Big)^{2.5\,k^{k+2}(2^k-1)+o\left(k^{k+2}(2^k-1)\right)} \cdot \log^3 n_1\right).$$

Hence, the problem of finding a largest stable matching in an instance of SPA-PUC is fixed parameter tractable when parameterised by the number of project topics k. □

3.5 Conclusions and future work

In this chapter, we presented complexity results for finding a maximum-size stable matching in restricted versions of SPA-ST and SPA-P. First, we showed that MAX-SPA-ST remains NP-hard even in the case where only one lecturer is involved. Then we showed that MAX-SPA-P remains NP-hard when both student and lecturer preferences are derived from a single master list of projects. On the other hand, we showed that MAX-SPA-P is polynomial-time solvable when each student finds acceptable only projects offered by a single lecturer. This was achieved by dividing the original instance into disjoint sub-instances of MAX-SPA-P-L1, for which a known polynomial-time algorithm can be applied. The final solution is then obtained by combining the solutions from each sub-instance. Additionally, we observed that MAX-SPA-P is solvable in polynomial time

when all students have identical preference lists. This follows from the fact that, in this scenario, all stable matchings are of the same size.

In addition to these results, we examined the parameterised complexity of MAX-SPA-P in a setting involving project topics and uniform capacities, denoted SPA-PUC. In this setting, students and lecturers express preferences over project topics rather than individual projects, and each lecturer, along with the projects they offer, has the same capacity. We showed that MAX-SPA-PUC is fixed-parameter tractable when parameterised by the number k of project topics. This was established by first applying two reductions to the original instance, resulting in an equivalent instance where the numbers of lecturers and projects are bounded in terms of the number of project topics. Then we formulated the problem as an ILP whose number of variables depends only on the number of project topics, k.

A possible direction for future work in the context of MAX-SPA-ST is to investigate whether the $\frac{3}{2}$-approximation algorithm by Cooper and Manlove [27] yields a better approximation factor in the case with a single lecturer. Another promising direction is to explore the parameterised complexity of this problem, particularly in a *typed* setting. We note that FPT algorithms have been developed for typed versions of MAX-SMTI and MAX-HRT [114]. Similar definitions of types could be developed for SPA-ST and examined further. In the SPA-P setting, the complexity of $(2, \infty)$-MAX-SPA-P, where each student ranks at most two projects and each lecturer is allowed an unbounded number of projects in their preference list, remains an open question. Future work in MAX-SPA-P could focus on tightening the parameterised bounds by exploring alternative structural parameters that lead to efficient algorithms. It would also be interesting to extend these results to other NP-hard variants of SPA-S, especially those involving additional constraints such as ties in preferences or lower quotas.

Chapter 4

Structural Results for SPA-S

4.1 Introduction

In this chapter, we study the structural properties of the Student-Project Allocation problem with lecturer preferences over Students (SPA-S). We give a new characterisation of the set of stable matchings for any instance of this problem. Our main result shows that these stable matchings form a distributive lattice, extending the well-known lattice structure from classical stable matching problems to this more general setting.

4.1.1 Background and motivation

As discussed in Sections 2.1.3.1 and 2.2.3, the set of stable matchings in the classical Stable Marriage problem (SM) and the Hospital Residents problem (HR) forms a distributive lattice. In these settings, the man-optimal (or resident-optimal) and woman-optimal (or hospital-optimal) stable matchings correspond to the minimum and maximum elements of the lattice, respectively. This structure has been central to the development of efficient algorithms for enumerating all stable matchings, identifying all stable pairs, and computing stable matchings that satisfy additional criteria, such as finding an egalitarian stable matching or a stable matching with minimum regret. Motivated by these applications, we investigate whether a similar structure exists in SPA-S, as this would enable the design of efficient algorithms for similar problems in the SPA-S model.

4.1.2 Contributions and structure of the chapter

We show that, for a given instance of the SPA-S problem, the set of all stable matchings forms a distributive lattice, with the student-optimal and lecturer-optimal stable matchings corresponding to the minimum and maximum elements of this lattice, respectively. A related result was previously established in Chapter 3 of [121], but under the restriction that each student provides preferences only over projects offered by different lecturers. In this chapter, we revisit some key results from that work, specifically Proposition 4.2.1 and Theorem 4.4.1. Additionally, we

develop new results and proofs to show that the distributive lattice structure also holds in the general case, without any restrictions on student preference lists.

4.2 Preliminary definitions

We refer the reader to the formal definitions of SPA-S presented in Section 2.3.1. To illustrate these definitions more concretely, we provide the following example:

Consider the SPA-S instance I_1 shown in Figure 4.1. Here, the set of students is $\mathcal{S} = \{s_1, s_2, \ldots, s_5\}$, the set of projects is $\mathcal{P} = \{p_1, p_2, \ldots, p_5\}$, and the set of lecturers is $\mathcal{L} = \{l_1, l_2\}$. Recall that each student has a preference list over the projects they find acceptable, and each lecturer ranks students in order of preference. In the example, s_1's preference list is p_1, p_2, and s_2's preference list is p_2, p_3. Also, lecturer l_1 offers p_1, p_2, p_5, while lecturer l_2 offers p_3, p_4. Moreover, l_1's preference list is s_4, s_5, s_3, s_1, s_2, and the projected preference list of l_1 for p_1 includes s_3, s_1, ranked in that order.

Students' preferences	**Lecturers' preferences**	**Offers**
s_1: p_1 p_2	l_1: s_4 s_5 s_3 s_1 s_2	p_1, p_2, p_5
s_2: p_2 p_3	l_2: s_2 s_3 s_5 s_4	p_3, p_4
s_3: p_3 p_1		
s_4: p_4 p_5		
s_5: p_5 p_4		
Project capacities: $c_1 = c_2 = c_3 = c_4 = c_5 = 1$		
Lecturer capacities: $d_1 = 3,\ d_2 = 2$		

Figure 4.1: An instance I_1 of SPA-S

With respect to the SPA-S instance I_1 shown in Figure 4.1, the matching $M_1 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_4), (s_5, p_5)\}$ is a stable matching, as it does not admit any blocking pair. Furthermore, M_1 is the *student-optimal* stable matching since every student is assigned to their best project in M_1. Similarly, the matching $M_2 = \{(s_1, p_2), (s_2, p_3), (s_3, p_1), (s_4, p_5), (s_5, p_4)\}$ is also stable in I_1, and M_2 is the *lecturer-optimal* stable matching. Clearly, in M_2, each lecturer whose assigned set of students differs from that in M_1, is assigned at least one student in M_2 whom they prefer to some student assigned to them in M_1.

4.2.1 Preferences over matchings

In this section, we extend the notion of preferences over individual projects (for students) and over individual students (for lecturers) to preferences over matchings. We then present the Unpopular Projects Theorem, originally introduced by [8] and presented in [121], which captures key structural properties of the set of stable matchings in SPA-S. Finally, we discuss how these properties differ from those in the HR model.

Theorem 4.2.1 (Unpopular Projects Theorem [8, 121])**.** *Let $\mathcal{M}$ denote the set of all stable matchings in a given instance of* SPA-S*. Then:*

(i) Each lecturer is assigned the same number of students in all stable matchings in $\mathcal{M}$.

(ii) Exactly the same students are unassigned in all stable matchings in $\mathcal{M}$.

(iii) Any project offered by an undersubscribed lecturer is assigned the same number of students in all stable matching in $\mathcal{M}$.

In the Rural Hospitals Theorem for the HR model (stated in 2.2.1), an undersubscribed hospital is assigned the same set of residents in every stable matching, and each hospital receives the same number of residents across all stable matchings. However, these properties do not fully extend to SPA-S. In particular:

- An undersubscribed lecturer may be assigned different sets of students in different stable matchings (see Figure 3 in [8]).
- A project offered by a full lecturer in one stable matching may be assigned a different number of students in another stable matching (see Figure 4 in [8]).

4.2.1.1 Student Preferences over Matchings

Let I be an instance of SPA-S, and let $\mathcal{M}$ denote the set of all stable matchings in I. Given two matchings $M, M' \in \mathcal{M}$, a student $s_i \in \mathcal{S}$ *prefers* M to M' if s_i is assigned in both matchings and prefers $M(s_i)$ to $M'(s_i)$. Similarly, s_i is *indifferent* between M and M' if either:

(i) s_i is unassigned in both M and M', or

(ii) s_i is assigned the same project in both matchings, i.e., $M(s_i) = M'(s_i)$.

4.2.1.2 Lecturer Preferences over Matchings

It is not immediately clear how to compare two stable matchings from the perspective of a lecturer. To formalise lecturer preferences over matchings, we adopt the definition proposed by

Abraham et al. in [8]. Let M and M' be two stable matchings in $\mathcal{M}$. By Theorem 4.2.1, $|M| = |M'|$ and $|M(l_k)| = |M'(l_k)|$ for each lecturer l_k. Suppose that l_k is assigned different sets of students in M and M'. Define

$$M(l_k) \setminus M'(l_k) = \{s_1, \ldots, s_r\}, \quad M'(l_k) \setminus M(l_k) = \{s'_1, \ldots, s'_r\},$$

where the students in each set are listed in the order they appear in l_k's preference list $\mathcal{L}_k$. Then l_k *prefers* M to M' if l_k prefers s_i to s'_i for all $i \in \{1, \ldots, r\}$. On the other hand, lecturer l_k is *indifferent* between M and M' if l_k is not assigned to any student or is assigned the same set of students in M and M', i.e., $M(l_k) = M'(l_k)$.

Example. Consider the two stable matchings M_1 and M_2 for instance I_1. Then:

$$M_2(l_1) \setminus M_1(l_1) = \{s_4, s_3\}, \quad M_1(l_1) \setminus M_2(l_1) = \{s_5, s_2\}.$$

The reader can verify that neither l_1 nor l_2 is assigned their most preferred set of students in both stable matchings. However, since l_1 prefers s_4 to s_5 and s_3 to s_2, it follows that l_1 prefers M_2 to M_1.

4.2.2 Dominance relation

We now define the dominance relation that plays a central role in constructing the lattice structure of stable matchings. Let $\mathcal{M}$ denote the set of all stable matchings in SPA-S. We show in Proposition 4.2.1 that $\mathcal{M}$, under the dominance relation $\preceq$, forms a partial order. Unless stated otherwise, whenever we write $M \preceq M'$, we refer to the *student-oriented* dominance relation. References to the lecturer-oriented dominance relation will be made explicit.

Definition 4.2.1 (Student-oriented dominance relation). Let $M, M' \in \mathcal{M}$. We say that M *dominates* M', denoted $M \preceq M'$, if and only if each student prefers M to M', or is indifferent between them.

Example. Consider instance I_1 in Figure 4.1, which admits the following two stable matchings: $M_1 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3), (s_4, p_4), (s_5, p_5)\}$, $M_2 = \{(s_1, p_2), (s_2, p_3), (s_3, p_1), (s_4, p_5), (s_5, p_4)\}$. Each student prefers their assignment in M_1 to their assignment in M_2, so M_1 dominates M_2.

Definition 4.2.2 (Lecturer-oriented dominance). Let $M, M' \in \mathcal{M}$. We say that M dominates M' from the lecturers' perspective if each lecturer either prefers M to M', or is indifferent between the two.

We note that in the hospital-resident setting, given any two stable matchings M and M', each hospital either prefers all of its assigned residents in M to those assigned to it in $M' \setminus M$, or prefers all its assigned residents in M' to those assigned to it in $M \setminus M'$. This property does

not hold in SPA-S. In SPA-S, if some lecturer l is assigned different sets of students in two stable matchings M and M', they may not prefer all students in $M(l)$ to those in $M'(l) \setminus M(l)$, nor all students in $M'(l)$ to those in $M(l) \setminus M'(l)$. However, it is always the case that l prefers at least one student in $M(l) \setminus M'(l)$ to at least one student in $M'(l) \setminus M(l)$, or vice versa.

In Figure 4.1, M_1 is the *student-optimal* stable matching since every student is assigned to their best project in M_1. Similarly, the matching $M_2 = \{(s_1, p_2), (s_2, p_3), (s_3, p_1), (s_4, p_5), (s_5, p_4)\}$ is also stable in I_1, and M_2 is the *lecturer-optimal* stable matching. Clearly, in M_2, each lecturer is assigned a student they prefer to at least one of the students assigned to them in M_1. In Lemma 4.3.1, we prove that for any two stable matchings M and M', if a student is assigned to lecturer l_k in both matchings, then there exists at least one student in $M'(l_k) \setminus M(l_k)$ and, consequently, one in $M(l_k) \setminus M'(l_k)$. We then prove in in Lemma 4.3.2 that if some student $s \in M(l_k) \setminus M'(l_k)$ prefers M to M', then l_k prefers M' to M.

Proposition 4.2.1. *Let $\mathcal{M}$ be the set of all stable matchings in I. The dominance relation $\preceq$ defines a partial order on $\mathcal{M}$, and we denote this partially ordered set as $(\mathcal{M}, \preceq)$.*

We remark that the proof given below follows a similar line of argument to that presented in [121].

Proof. We show that the dominance relation $\preceq$ on $\mathcal{M}$ is: (i) reflexive, (ii) anti-symmetric, and (iii) transitive.

(i) **Reflexive:** Let $M \in \mathcal{M}$. Clearly, $M \preceq M$, since every student is indifferent between M and itself. Thus, $\preceq$ is reflexive.

(ii) **Anti-symmetric:** Let $M, M' \in \mathcal{M}$ such that $M \preceq M'$ and $M' \preceq M$. Then $M = M'$. Suppose, for contradiction, that $M \neq M'$. Then there exists some student s_i such that s_i is assigned in both M and M', and $M(s_i) \neq M'(s_i)$. Since $M \preceq M'$, s_i prefers $M(s_i)$ to $M'(s_i)$. Similarly, $M' \preceq M$ implies s_i prefers $M'(s_i)$ to $M(s_i)$. This is a contradiction. Hence, $M = M'$, and $\preceq$ on $\mathcal{M}$ is anti-symmetric.

(iii) **Transitive:** Let $M, M', M'' \in \mathcal{M}$ such that $M \preceq M'$ and $M' \preceq M''$. We claim that $M \preceq M''$. By Theorem 4.2.1, we know that exactly the same students are unassigned in all stable matchings. Thus, every student who is unassigned in M is unassigned in M'', and every unassigned student is indifferent between M and M''. Clearly, every student who is assigned to the same project in M and M'' is indifferent between M and M''.

Now, let s_i be some student who is assigned to different projects in both M and M'', say $M(s_i)$ and $M''(s_i)$ respectively. First, suppose that $M(s_i) \neq M'(s_i)$; since $M \preceq M'$, it follows that s_i prefers $M(s_i)$ to $M'(s_i)$. Further, (a) if $M'(s_i) = M''(s_i)$ then s_i prefers $M(s_i)$ to $M''(s_i)$, and (b) if $M'(s_i) \neq M''(s_i)$, $M' \preceq M''$ implies that s_i prefers $M'(s_i)$ to $M''(s_i)$,

and since the preference lists are strictly ordered, s_i prefers $M(s_i)$ to $M''(s_i)$. Now, suppose that $M'(s_i) = M''(s_i)$. It follows that $M'(s_i) \neq M''(s_i)$; thus $M' \preceq M''$ implies that s_i prefers $M'(s_i)$ to $M''(s_i)$. This implies that s_i prefers $M(s_i)$ to $M''(s_i)$. Hence our claim holds; and therefore $\preceq$ on $\mathcal{M}$ is transitive.

□

Definition 4.2.3 (Distributive lattice [54]**).** Let A be a set and let $\preceq$ be an ordering relation defined on A. The partial order $(A, \preceq)$ is a distributive lattice if:

(i) each pair of element $x, y \in A$ has a greatest lower bound, or meet, denoted $x \wedge y$, such that $x \wedge y \preceq x$, $x \wedge y \preceq y$, and there is no element $z \in A$ for which $z \preceq x$, $z \preceq y$ and $x \wedge y \preceq z$;

(ii) each pair of element $x, y \in A$ has a least upper bound, or join, denoted $x \vee y$, such that $x \preceq x \vee y$, $y \preceq x \vee y$, and there is no element $z \in A$ for which $x \preceq z$, $y \preceq z$ and $z \preceq x \vee y$;

(iii) the join and meet distribute over each other, i.e., for $x, y, z \in A$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ and $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

4.3 Structural properties of stable matchings

In this section, we present new results that illustrate lecturers' preferences over matchings when a student prefers one stable matching to another. These results will be used in the next section to prove that the set of stable matchings forms a distributive lattice. We first present Proposition 4.3.1, which is used in the proofs of Lemmas 4.3.1 and 4.3.2.

Let M and M' be two stable matchings in a SPA-S instance I. In Lemma 4.3.1, we show that if a student s_i is assigned to different projects offered by the same lecturer l_k in M and M', and s_i prefers M to M', then l_k prefers some student in $M'(l_k) \setminus M(l_k)$ to s_i. In Lemma 4.3.2, we show that if there exists a student $s \in M(l_k) \setminus M'(l_k)$ who prefers M to M', then l_k prefers M' to M. In Lemma 4.3.3, we show that if s_i is assigned to p_j offered by l_k in M' and prefers M to M', then l_k prefers s_i to each student in $M(p_j) \setminus M'(p_j)$, or, if p_j is undersubscribed in M, to each student in $M(l_k) \setminus M'(l_k)$. Finally, in Lemma 4.3.4, we prove the symmetric case of Lemma 4.3.1: if s_i is assigned to different projects offered by the same lecturer l_k, then l_k prefers s_i to some student in $M(l_k) \setminus M'(l_k)$.

Proposition 4.3.1. *Let M and M' be two stable matchings in I, and let s be some student assigned in M to a project p_j offered by lecturer l_k. If s prefers M to M' and either $s \in M'(l_k)$ or l_k prefers s to at least one student in $M'(l_k)$, then p_j is full in M'.*

Proof. Let s be some student assigned in M to p_j offered by l_k, where s prefers M to M'. Suppose, for contradiction, that p_j is undersubscribed in M'. Then, if $s \in M'(l_k)$ or l_k prefers s to some student in $M'(l_k)$, it follows that (s, p_j) forms a blocking pair in M'. This contradicts the stability of M'. Hence, p_j is full in M' and our claim holds. □

Lemma 4.3.1. *Let M and M' be two stable matchings in I. If some student s_i is assigned in M and M' to different projects offered by the same lecturer l_k, and s_i prefers M to M', then there exists some other student $s_r \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s_r to s_i. Thus, $M(l_k) \neq M'(l_k)$.*

Proof. Let M and M' be two stable matchings in I. Let s_i be some student assigned to different projects in M and M', both offered by the same lecturer l_k, and suppose s_i prefers M to M'. Suppose for contradiction that there exists no student $s_r \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s_r to s_i. Then by the Unpopular Projects Theorem 4.2.1, it follows that $M(l_k) = M'(l_k)$, i.e., $M'(l_k) \setminus M(l_k) = \emptyset$. We construct sequences $\langle s_1, s_2, s_3, \ldots \rangle$ and $\langle p_0, p_1, p_2, \ldots \rangle$ of students and projects such that for each $t \geq 2$:

(1) s_t prefers p_t to p_{t-1},

(2) $(s_t, p_t) \in M \setminus M'$ and $(s_t, p_{t-1}) \in M' \setminus M$,

(3) l_k offers both p_t and p_{t-1},

(4) l_k prefers s_t to s_{t-1}.

We prove by induction that these properties hold for all $t \geq 2$. Let $s_1 = s_i$.

Base case ($t = 2$). Let $p_1 = M(s_1)$, $p_0 = M'(s_1)$. Since s_1 prefers M to M', it follows that s_1 prefers p_1 to p_0. Moreover, $(s_1, p_1) \in M \setminus M'$, $(s_1, p_0) \in M' \setminus M$, and both projects are offered by l_k. Since M' is stable, one of the following conditions hold:

(i) p_1 is full in M', and l_k prefers the worst student in $M'(p_1)$ to s_1; or

(ii) p_1 is undersubscribed in M', l_k is full in M', $s_1 \notin M'(l_k)$, and l_k prefers the worst student in $M'(l_k)$ to s_1.

Since $s_1 \in M'(l_k)$, it follows from Proposition 4.3.1 that p_1 is full in M' and case (i) holds. In this case, since $(s_1, p_1) \in M \setminus M'$, there exists some student $s_2 \in M'(p_1) \setminus M(p_1)$; otherwise, p_1 would be oversubscribed in M. Furthermore, l_k prefers s_2 to s_1. Now, in M, s_2 must be assigned to some project p_2 such that s_2 prefers p_2 to p_1; otherwise, (s_2, p_1) would block M. Hence $(s_2, p_2) \in M \setminus M'$. Moreover, since $s_2 \in M'(l_k)$, and we assumed $M(l_k) = M'(l_k)$, it follows that l_k also offers p_2. Thus, properties (1)–(4) hold for $t = 2$, completing the base case.

Inductive step. Assume that properties (1) - (4) above hold for some $t = q - 1 \geq 2$. We now show that the properties also hold for $t = q$. By the inductive hypothesis:

(1) s_{q-1} prefers p_{q-1} to p_{q-2},

(2) $(s_{q-1}, p_{q-1}) \in M \setminus M'$, $(s_{q-1}, p_{q-2}) \in M' \setminus M$,

(3) l_k offers both p_{q-1} and p_{q-2}.

(4) l_k prefers s_{q-1} to s_{q-2},

Since M' is stable, one of the following conditions hold:

(i) p_{q-1} is full in M', and l_k prefers the worst student in $M'(p_{q-1})$ to s_{q-1};

(ii) p_{q-1} is undersubscribed in M', l_k is full, $s_{q-1} \notin M'(l_k)$, and l_k prefers the worst student in $M'(l_k)$ to s_{q-1}.

Since $s_{q-1} \in M'(l_k)$, it follows from Proposition 4.3.1 that p_{q-1} is full in $M'$and case (i) holds. Since $(s_{q-1}, p_{q-1}) \in M \setminus M'$, there exists some student, say s_q, such that $(s_q, p_{q-1}) \in M' \setminus M$; otherwise, p_{q-1} would be oversubscribed in M. Furthermore, l_k prefers s_q to s_{q-1}. Now, in M, s_q must be assigned to some project p_q such that s_q prefers p_q to p_{q-1}; otherwise (s_q, p_{q-1}) would block M. Hence, $(s_q, p_q) \in M \setminus M'$. Also, since $s_q \in M'(l_k)$ and $M(l_k) = M'(l_k)$, it follows that l_k also offers p_q. Thus, properties (1) - (4) hold for $t = q$, completing the inductive step.

It is easy to see that for each new student that we identify, l_k prefers s_t to s_{t-1} and prefers s_{t-1} to s_{t-2}, ..., and prefers s_2 to s_1, just as in Figure 4.2. Hence, all identified students must be distinct. Since the sequence of distinct students is infinite, we reach an immediate contradiction. This contradiction implies that $M(l_k) \neq M'(l_k)$, and the sequence must terminate with some student $s_r \in M'(l_k) \setminus M(l_k)$. Hence, $M(l_k) \neq M'(l_k)$, as required.

	M	M'								
s_1 :	p_1	p_0	l_k:	...	s_t	s_{t-1}	...	s_3	s_2	s_1
s_2 :	p_2	p_1								
s_3 :	p_3	p_2								
⋮	⋮	⋮								
s_{t-1} :	p_{t-1}	p_{t-2}								
s_t :	p_t	p_{t-1}								
⋮	⋮	⋮								

Figure 4.2: An illustration of the sequence of students generated in Lemma 4.3.1, with $(s_r, p_r) \in M$ and $(s_r, p_{r-1}) \in M'$ for all $r \geq 2$

□

As an example, consider the instance I_1 in Figure 4.1. Here, s_1 is assigned to p_1 in M_1 and p_2 in M_2, where both p_1 and p_2 are offered by l_1. By Lemma 4.3.1, we can identify some $s_3 \in M_2(l_1) \setminus M_1(l_1)$ such that l_1 prefers s_3 to s_1.

Lemma 4.3.2. *Let M and M' be two stable matchings in an instance I, and let l_k be a lecturer such that $M(l_k) \neq M'(l_k)$. If there exists a student $s \in M(l_k) \setminus M'(l_k)$ who prefers M to M', then l_k prefers M' to M.*

Proof. Let M and M' be two stable matchings in $\mathcal{M}$. Let l_k be some lecturer such that $M(l_k) \neq M'(l_k)$, and let $s_i \in M(l_k) \setminus M'(l_k)$ be some student who prefers M to M'. To prove that l_k prefers M' to M, we construct a one-to-one mapping

$$f : M(l_k) \setminus M'(l_k) \rightarrow M'(l_k) \setminus M(l_k)$$

such that for each student $s \in M(l_k) \setminus M'(l_k)$ who prefers M to M', l_k prefers $f(s)$ to s. That is, for each such student in $M(l_k) \setminus M'(l_k)$, we can find a corresponding student $s' \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s' to s.

We say that a student $s_x \in M(l_k)$ is a *dominated student* if l_k prefers all students in $M'(l_k)$ to s_x. There are two possible cases for the students in $M(l_k)$ who prefer M to M':

Case 1: All such students are dominated.
In this case, l_k prefers all students in $M'(l_k)$ to each such student in $M(l_k) \setminus M'(l_k)$. Since $|M'(l_k) \setminus M(l_k)| = |M(l_k) \setminus M'(l_k)|$, we can construct a one-to-one mapping for each student $s \in M(l_k) \setminus M'(l_k)$ who prefers M to M' to another student $s' \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s' to s. In this way, the mapping is valid, and l_k prefers M' to M.

Case 2: There exists at least one student in $M(l_k) \setminus M'(l_k)$ who prefers M to M' and is not dominated.
Let $s_1 \in M(l_k) \setminus M'(l_k)$ be a non-dominated student who prefers M to M', and let $p_1 = M(s_1)$. It follows that l_k prefers s_1 to at least one student in $M'(l_k)$. Since M' is a stable matching, then either (i) or (ii) holds as follows:

(i) p_1 is full in M', and l_k prefers the worst student in $M'(p_1)$ to s_1; or

(ii) p_1 is undersubscribed in M', l_k is full in M', $s_1 \notin M'(l_k)$, and l_k prefers each student in $M'(l_k)$ to s_1.

Since l_k prefers s_1 to at least one student in $M'(l_k)$, it follows from Proposition 4.3.1 that p_1 is full in M'. So case (i) holds, and there exists some student $s_2 \in M'(p_1) \setminus M(p_1)$ such that l_k prefers s_2 to s_1. First suppose that s_2 prefers M' to M. If p_1 is full in M, then l_k prefers s_2 to some student in $M(p_1)$ (namely s_1), so (s_2, p_1) blocks M. If p_1 is undersubscribed in M, then l_k prefers s_2 to some student in $M(l_k)$ (namely s_1), and (s_2, p_1) again blocks M (this holds whether l_k is full or undersubscribed in M). Therefore, s_2 prefers M to M'.

Let $S_k(M, M')$ denote the set of students assigned in M and M' to different projects offered by lecturer l_k. If $s_2 \in M'(l_k) \setminus M(l_k)$, we define $f(s_1) = s_2$ and stop. Otherwise, $s_2 \in S_k(M, M')$. In this case, let $p_2 = M(s_2)$. Then p_2 is offered by l_k and $(s_2, p_2) \in M \setminus M'$. Since M' is a stable matching, then either (i) or (ii) holds as follows:

(i) p_2 is full in M', and l_k prefers the worst student in $M'(p_2)$ to s_2; or

(ii) p_2 is undersubscribed in M', l_k is full in M', $s_2 \notin M'(l_k)$, and l_k prefers each student in $M'(l_k)$ to s_2.

Since l_k prefers s_2 to s_1, and prefers s_1 to at least one student in $M'(l_k)$, then l_k prefers s_2 to at least one student in $M'(l_k)$. By Proposition 4.3.1, it follows that p_2 is full in M'. Since $s_2 \in M(p_2) \setminus M'(p_2)$ and p_2 is full in M', there exists some student $s_3 \in M'(p_2) \setminus M(p_2)$; for otherwise, p_2 is oversubscribed in M. Moreover, l_k prefers s_3 to s_2. Suppose that s_3 prefers M' to M. If p_2 is full in M, then l_k prefers s_3 to some student in $M(p_2)$ (namely s_2), so (s_3, p_2) blocks M. If p_2 is undersubscribed in M, then l_k prefers s_3 to some student in $M(l_k)$ (namely s_2), and (s_3, p_2) again blocks M. Therefore, s_3 prefers M to M'. If $s_3 \in M'(l_k) \setminus M(l_k)$, we define $f(s_1) = s_3$ and stop. Otherwise, we continue this process to obtain a sequence of students $s_4, s_5, \ldots, s_t$, where each student in the sequence prefers M to M' and is preferred by l_k to their predecessor; that is, l_k prefers s_r to s_{r-1} for $1 < r \leq t$. Since the number of students is finite, this sequence must eventually terminate with a student $s_t \in M'(l_k) \setminus M(l_k)$, at which point we define $f(s_1) = s_t$.

The sequence $s_1, s_2, \ldots, s_t$ is such that

- $s_1 \in M(l_k) \setminus M'(l_k)$
- $s_r \in M(l_k) \cap M'(l_k)$ for $1 < r < t$
- $s_t \in M'(l_k) \setminus M(l_k)$
- l_k prefers s_r to s_{r-1} for $1 < r \leq t$.
- s_r prefers M to M' for $1 \leq r \leq t$.

We repeat this construction for every non-dominated student in $M(l_k) \setminus M'(l_k)$. We also ensure that if the same project appears in the sequences starting from multiple students in $M(l_k) \setminus M'(l_k)$, then we can assign a distinct student from $M'(l_k) \setminus M(l_k)$ to that project in each case. Suppose some project $p_x \in P_k$ arises multiple times in $M \setminus M'$, each time assigned to a different student. Then, as argued earlier, p_x must be full in M', and all students assigned to p_x in M' are preferred by l_k to the students it was assigned to in M. Since each occurrence of p_x corresponds to a different student in $M(l_k) \setminus M'(l_k)$, and p_x is full in M', there are sufficiently many students in $M'(l_k) \setminus M(l_k)$ assigned to p_x from which we can choose. We select a distinct one for each occurrence, preserving the one-to-one mapping.

Finally, for the dominated students, we assign the remaining unassigned students in $M'(l_k) \setminus M(l_k)$ arbitrarily. Since for each dominated student $s \in M(l_k) \setminus M'(l_k)$, l_k prefers each student in $M'(l_k)$ to s, the condition that l_k prefers $f(s)$ to s still holds. Thus, in both cases, we construct a valid one-to-one mapping from $M(l_k) \setminus M'(l_k)$ to $M'(l_k) \setminus M(l_k)$ such that l_k prefers each student in $M'(l_k) \setminus M(l_k)$ to the corresponding student in $M(l_k) \setminus M'(l_k)$. Therefore, l_k prefers M' to M, as required. □

Lemma 4.3.3. *Let M and M' be two stable matchings in a given instance I. Suppose some student s_i is assigned to different projects in M and M', and that in M', s_i is assigned to a project p_j offered by lecturer l_k. Suppose further that s_i prefers M to M'. Then:*

(a) If there exists a student in $M(p_j) \setminus M'(p_j)$, then l_k prefers s_i to each student in $M(p_j) \setminus M'(p_j)$.

(b) If p_j is undersubscribed in M, then l_k prefers s_i to each student in $M(l_k) \setminus M'(l_k)$.

Proof. Let M and M' be two stable matchings in I, and let s_i be a student assigned to different projects in M and M', where s_i prefers M to M'. Let $p_j = M'(s_i)$, and let l_k be the lecturer offering p_j. In Case (a), we show that if $M(p_j) \setminus M'(p_j)$ is non-empty, then there exists a student $s' \in M(p_j) \setminus M'(p_j)$ who prefers M' to M. In Case (b), we show that if p_j is undersubscribed in M, then there exists a student $s' \in M(l_k)$ who is assigned to different projects in M and M', and who prefers M' to M. In both cases, we use the student s' identified to initiate an inductive argument. This produces a sequence of distinct students, where each student prefers M' to M, and is preferred, by the lecturer they are assigned to in M, to the previous student in the sequence.

Case (a): Suppose there exists some student $s' \in M(p_j) \setminus M'(p_j)$, and suppose for contradiction that l_k prefers s' to s_i. Suppose p_j is full in M'. If s' prefers M to M', then (s', p_j) blocks M', since l_k prefers s' to some student in $M'(p_j)$ (namely s_i), a contradiction. Now suppose that p_j is undersubscribed in M'. Since l_k prefers s' to some student in $M'(l_k)$ (again, s_i), then (s', p_j) blocks M', another contradiction. Therefore, s' prefers M' to M.

Case (b): Suppose p_j is undersubscribed in M, and suppose for contradiction that l_k prefers each student in $M(l_k) \setminus M'(l_k)$ to s_i. Recall that $s_i \in M'(p_j) \setminus M(p_j)$. Since p_j is undersubscribed in M and $|M(l_k)| = |M'(l_k)|$, there exists some project $p' \in P_k$ and some student $s' \in M(p') \setminus M'(p')$, where p' is undersubscribed in M'. First suppose that s' prefers M to M'. If $s' \in S_k(M, M')$, then (s', p') blocks M', a contradiction. Thus, $s' \in M(l_k) \setminus M'(l_k)$. Since, by our assumption, l_k prefers each student in $M(l_k) \setminus M'(l_k)$ to s_i, it follows that l_k prefers s' to s_i. However, the pair (s', p') again blocks M'. Therefore, s' prefers M' to M.

The remainder of the proof for Cases (a) and (b) proceeds in an identical manner. We therefore continue with the inductive step that satisfies the conditions of both cases in the following paragraph.

Let s' be the student, identified in either Case (a) or Case (b) above, who prefers M' to M. Let $s_0 = s_i$, $l_0 = l_k$. Let $s_1 = s'$, $p_0 = M(s_1)$, and $p_1 = M'(s_1)$, and let l_1 be the lecturer who offers p_1. Note that it is possible that $p_0 = p_j$ and $l_1 = l_0$. We have $(s_1, p_1) \in M' \setminus M$, $(s_1, p_0) \in M \setminus M'$, and l_0 prefers s_1 to s_0. Moreover, s_1 prefers M' to M, while s_0 prefers M to M'.

We now proceed by identifying students $s_2, s_3, \ldots$, projects $p_2, p_3, \ldots$, and lecturers $l_2, l_3, \ldots$, such that for each $t \geq 2$, the following properties hold:

(1) s_t prefers M' to M;

(2) $(s_t, p_t) \in M' \setminus M$, where project p_t is offered by lecturer l_t;

(3) s_t is assigned in M to some project offered by lecturer l_{t-1}, and:

- if p_{t-1} is full in M, then $(s_t, p_{t-1}) \in M \setminus M'$;
- otherwise, there exists a project p'_{t-1} such that $(s_t, p'_{t-1}) \in M \setminus M'$ (where l_{t-1} offers both p_{t-1} and p'_{t-1}).

(4) Lecturer l_{t-1} prefers s_t to s_{t-1}.

Base case (t = 2): Since s_1 prefers M' to M, so s_1 prefers p_1 to p_0. Moreover, since $(s_1, p_1) \in M' \setminus M$ and $(s_1, p_0) \in M \setminus M'$, it follows that $p_1 \neq p_0$. Also, lecturer l_0 prefers s_1 to s_0. By the stability of M, one of the following two cases holds:

(i) p_1 is full in M, and l_1 prefers the worst student in $M(p_1)$ to s_1;

(ii) p_1 is undersubscribed in M, l_1 is full in M, $s_1 \notin M(l_1)$, and l_1 prefers the worst student in $M(l_1)$ to s_1.

Case (i): Since p_1 is full in M and $(s_1, p_1) \notin M$, there exists another student s_2 such that $(s_2, p_1) \in M \setminus M'$, otherwise p_1 would be oversubscribed in M. Moreover, l_1 prefers s_2 to s_1. Clearly, s_2 is assigned in M'; let $p_2 = M'(s_2)$, and let l_2 be the lecturer who offers p_2. Then $(s_2, p_2) \in M' \setminus M$. If s_2 prefers p_1 to p_2, then the pair (s_2, p_1) blocks M', since p_1 is full in M, and l_1 prefers s_2 to $s_1 \in M'(p_1)$. Thus, s_2 must prefer p_2 to p_1, that is, s_2 prefers M' to M. Note that $s_2 \neq s_1$, since l_1 prefers s_2 to s_1; and $s_2 \neq s_0$, since s_0 prefers M to M', while s_2 prefers M' to M.

Case (ii): Since $(s_1, p_1) \in M' \setminus M$ and p_1 is undersubscribed in M, there exists some project p'_1 offered by l_1, such that p'_1 is undersubscribed in M'; otherwise, l_1 would be oversubscribed in M. Thus, there exists some student s_2 such that $(s_2, p'_1) \in M \setminus M'$ and l_1 prefers s_2 to s_1. Moreover, s_2 is assigned in M'; let $p_2 = M'(s_2)$, and let l_2 be the lecturer who offers p_2. Then $(s_2, p_2) \in M' \setminus M$. If s_2 prefers p'_1 to p_2, then the pair (s_2, p'_1) would block M', since p'_1 is undersubscribed in M' and l_1 prefers s_2 to s_1. Hence, s_2 prefers p_2 to p'_1, i.e., s_2 prefers M' to M. As before, $s_2 \neq s_1$, since l_1 prefers s_2 to s_1, and $s_2 \neq s_0$, since s_2 prefers M' to M, whereas s_0 prefers M to M'.

In both cases (i) and (ii), we have identified a student s_2 who prefers M' to M, with $(s_2, p_2) \in M' \setminus M$, where p_2 is offered by lecturer l_2. Moreover, s_2 is assigned in M to some project offered by l_1. If p_1 is full in M, then $(s_2, p_1) \in M \setminus M'$; otherwise, $(s_2, p'_1) \in M \setminus M'$. Moreover, l_1 prefers s_2 to s_1. Thus, properties (1)–(4) hold for $t = 2$, completing the base case.

Inductive step: Suppose properties (1)-(4) hold for some $t = q - 1 \geq 2$, that is:

(1) s_{q-1} prefers M' to M;

(2) $(s_{q-1}, p_{q-1}) \in M' \setminus M$, where p_{q-1} is offered by lecturer l_{q-1};

(3) s_{q-1} is assigned in M to some project offered by lecturer l_{q-2}, and:

- if p_{q-2} is full in M, then $(s_{q-1}, p_{q-2}) \in M \setminus M'$;
- otherwise, there exists a project p'_{q-2} such that $(s_{q-1}, p'_{q-2}) \in M \setminus M'$ (where l_{q-2} offers both p_{q-2} and p'_{q-2}).

(4) Lecturer l_{q-2} prefers s_{q-1} to s_{q-2}.

By the stability of M, one of the following two cases must hold:

(i) p_{q-1} is full in M, and l_{q-1} prefers the worst student in $M(p_{q-1})$ to s_{q-1};

(ii) p_{q-1} is undersubscribed in M, l_{q-1} is full in M, $s_{q-1} \notin M(l_{q-1})$, and l_{q-1} prefers the worst student in $M(l_{q-1})$ to s_{q-1}.

Case (i): Since p_{q-1} is full in M and $(s_{q-1}, p_{q-1}) \notin M$, there exists a student s_q such that $(s_q, p_{q-1}) \in M \setminus M'$, otherwise p_{q-1} would be oversubscribed in M. Moreover, l_{q-1} prefers s_q to s_{q-1}. Clearly, s_q is assigned in M'; let $p_q = M'(s_q)$, and let l_q be the lecturer who offers p_q. Then $(s_q, p_q) \in M' \setminus M$. If s_q prefers p_{q-1} to p_q, then the pair (s_q, p_{q-1}) blocks M', since p_{q-1} is full in M and l_{q-1} prefers s_q to $s_{q-1} \in M'(p_{q-1})$. Hence s_q prefers p_q to p_{q-1}, that is, s_q prefers M' to M.

Case (ii): Since $(s_{q-1}, p_{q-1}) \in M' \setminus M$ and p_{q-1} is undersubscribed in M, there must exist some project $p'_{q-1} \in P_{q-1}$ that is undersubscribed in M', for otherwise l_{q-1} would be oversubscribed in M. Then there exists a student s_q such that $(s_q, p'_{q-1}) \in M \setminus M'$, and l_{q-1} prefers s_q to s_{q-1}. Moreover, s_q is assigned in M'; let $p_q = M'(s_q)$ and let l_q be the lecturer who offers p_q. Then $(s_q, p_q) \in M' \setminus M$. If s_q prefers p'_{q-1} to p_q, then the pair (s_q, p'_{q-1}) blocks M', since p'_{q-1} is undersubscribed in M' and l_{q-1} prefers s_q to $s_{q-1} \in M'(l_{q-1})$. Hence s_q prefers p_q to p'_{q-1}, that is, s_q prefers M' to M.

In both cases (i) and (ii), we have identified a student s_q who prefers M' to M, with $(s_q, p_q) \in M' \setminus M$, where p_q is offered by lecturer l_q. Moreover, s_q is assigned in M to some project offered by l_{q-1}. If p_{q-1} is full in M, then $(s_q, p_{q-1}) \in M \setminus M'$; otherwise, $(s_q, p'_{q-1}) \in M \setminus M'$. Moreover, l_{q-1} prefers s_q to s_{q-1}. Thus, properties (1)–(4) hold for $t = q$, completing the base case.

It follows from the construction that each project p_t differs from the previous project p_{t-1}, since each student s_t is assigned to different projects in M and M'. We now show that all students in the sequence $s_1, s_2, \ldots$ are distinct, by induction on t. Clearly, $s_2 \neq s_1$, since the lecturer l_1 prefers s_2 to s_1, and $s_2 \neq s_0$, since s_2 prefers M' to M while s_0 prefers M to M'. Now suppose, as the inductive hypothesis, that $s_1, \ldots, s_t$ are all distinct for some $t \geq 2$, and suppose for contradiction that $s_{t+1} = s_q$ for some $1 \leq q \leq t$. In the construction, s_{t+1} is selected by lecturer l_t to prevent the pair $(s_t, p_t) \in M' \setminus M$ from blocking M. This means that either $(s_{t+1}, p_t) \in M \setminus M'$, if p_t is full in M, or $(s_{t+1}, p'_t) \in M \setminus M'$, if p_t is undersubscribed in M. In both cases, l_t prefers s_{t+1} to s_t.

Since $s_{t+1} = s_q$, this student must have appeared earlier in the sequence and must have been selected to resolve a different blocking pair involving some earlier student s_{q-1}. In that case, l_t is using the same student s_q to resolve two different blocking pairs: one involving (s_t, p_t) and one involving (s_{q-1}, p_t). But by the inductive hypothesis, $s_t \neq s_{q-1}$, and so l_t should have selected two different students. This contradicts the construction, which requires that each blocking pair is resolved by a student who has not already appeared in the sequence. Therefore, $s_{t+1} \neq s_q$ for all $1 \leq q \leq t$, and so the sequence $s_1, s_2, \ldots$ consists of distinct students. Since the number of students is finite, the construction must eventually terminate. This establishes our claim and completes the proof.

To illustrate this proof, consider Figure 4.3. Suppose that a student s_0 prefers M to M', where in M', s_0 is assigned to p_0, a project offered by l_0. Further, suppose there exists a student $s_1 \in M(p_0) \setminus M'(p_0)$, and suppose for a contradiction that l_0 prefers s_1 to s_0. Clearly, if s_1 also prefers M to M', then (s_1, p_0) blocks M'. This implies that s_1 prefers M' to M. Let $M'(s_1)$ be p_1, where l_1 offers p_1. To ensure the stability of M, we continue identifying a sequence of distinct students, as illustrated in Figure 4.3. However, since this sequence is infinite, we arrive at a contradiction.

Students' preferences	Lecturers' preferences	Offers
s_0: p'_0 p_0	l_0: s_1 s_0	p_0
s_1: p_1 p_0	l_1: s_2 s_1	p_1
s_2: p_2 p_1	l_2: s_3 s_2	p_2
s_3: p_3 p_2	l_3: s_4 s_3	p_3
$\vdots$ $\vdots$ $\vdots$	$\vdots$	$\vdots$
s_t: p_t p_{t-1}	l_t: s_{t+1} s_t	p_t
$\vdots$ $\vdots$ $\vdots$	$\vdots$	$\vdots$

Figure 4.3: A SPA-S instance illustrating the infinite sequence of students generated in Lemma 4.3.3, where $(s_t, p_t) \in M'$ and $(s_t, p_{t-1}) \in M$.

□

Lemma 4.3.4. *Let M and M' be two stable matchings in I. If some student s_i is assigned in M and M' to different projects offered by the same lecturer l_k, and s_i prefers M to M', then l_k prefers s_i to some student $s_z \in M(l_k) \setminus M'(l_k)$.*

Proof. Let M and M' be two stable matchings in I, and let s_1 be a student assigned to different projects in M and M', both offered by lecturer l_k, where s_1 prefers M to M'. By Lemma 4.3.1, there exists a student in $M'(l_k) \setminus M(l_k)$ and, consequently, one in $M(l_k) \setminus M'(l_k)$. Suppose, for a contradiction, that no student $s \in M(l_k) \setminus M'(l_k)$ is worse than s_1 according to l_k. Let $M(s_1) = p_0$ and $M'(s_1) = p_1$, where s_1 prefers p_0 to p_1. If p_1 is undersubscribed in M, then by the second part of Lemma 4.3.3, l_k prefers s_1 to each student in $M(l_k) \setminus M'(l_k)$, a contradiction. Hence p_1 must be full in M. Since $(s_1, p_1) \in M' \setminus M$ and p_1 is full in M, there exists some $(s_2, p_1) \in M \setminus M'$, and by the first part of Lemma 4.3.3, l_k prefers s_1 to s_2.

If $s_2 \in M(l_k) \setminus M'(l_k)$, then l_k prefers s_1 to some student in $M(l_k) \setminus M'(l_k)$, contradicting our assumption. Hence $s_2 \in S_k(M, M')$. Let $M'(s_2) = p_2$. First suppose that s_2 prefers p_2 to p_1, i.e. s_2 prefers M' to M. Since $s_1 \in M'(p_1) \setminus M(p_1)$, part (a) of Lemma 4.3.3 implies that l_k prefers s_2 to s_1, a contradiction. Thus s_2 prefers p_1 to p_2. If p_2 is undersubscribed in M, then by Lemma 4.3.3, l_k prefers s_2 to each student in $M(l_k) \setminus M'(l_k)$. Since l_k prefers s_1 to s_2, it follows that l_k prefers s_1 to student $s \in M(l_k) \setminus M'(l_k)$, which again contradicts our assumption. Hence p_2 is full in M. Since $(s_2, p_2) \in M' \setminus M$ and p_2 is full in M, there exists $(s_3, p_2) \in M \setminus M'$, and by the first part of Lemma 4.3.3, l_k prefers s_2 to s_3.

Again, if $s_3 \in M(l_k) \setminus M'(l_k)$, then l_k prefers s_1 to s_2, and s_2 to some student in $M(l_k) \setminus M'(l_k)$, contradicting our assumption. Thus $s_3 \in S_k(M, M')$, and let $M'(s_3) = p_3$. Proceeding inductively as before, we obtain a sequence $s_1, s_2, s_3, \ldots$ such that, for each $t \geq 1$, s_t prefers p_{t-1} to p_t, $(s_t, p_{t-1}) \in M \setminus M'$, $(s_t, p_t) \in M' \setminus M$, and both p_{t-1} and p_t are offered by l_k, who prefers s_t to s_{t+1}. Hence l_k prefers s_1 to s_2, s_2 to s_3, and so on, implying that all identified students are distinct. Since the number of students is finite, this sequence cannot continue indefinitely and must terminate with some $s \in M(l_k) \setminus M'(l_k)$ such that l_k prefers s_1 to s. □

The following corollary follows from Lemmas 4.3.2 - 4.3.4:

Corollary 4.3.1. *Let M and M' be stable matchings in an instance I, and let $\preceq_S$ and $\preceq_L$ denote the student-oriented and lecturer-oriented dominance relations, respectively. Then $M \preceq_S M'$ if and only if $M' \preceq_L M$.*

Proof. ($\Rightarrow$) Suppose $M \preceq_S M'$. Then each student either prefers M to M' or is indifferent between them. If $M = M'$, the claim is immediate. Otherwise, consider any lecturer l_k. If $M(l_k) = M'(l_k)$, then l_k is indifferent between M and M', so $M' \preceq_L M$ holds for l_k. If $M(l_k) \neq M'(l_k)$, then there exist some student $s \in M(l_k) \setminus M'(l_k)$. Since $M \preceq_S M'$, s prefers M to M'. By Lemma 4.3.2, it follows that l_k prefers M' to M. Thus, for every l_k, l_k is either indifferent or prefers M' to M, i.e., $M' \preceq_L M$.

($\Leftarrow$) Conversely, suppose $M' \preceq_L M$. Then each lecturer either prefers M' to M or is indifferent between them. Consider any student s. If $M(s) = M'(s)$, then s is indifferent between the two matchings, so $M \preceq_S M'$ holds for s. Otherwise, let l_k be the lecturer offering $M'(s)$, and suppose, for a contradiction, that s prefers M' to M. If $s \in S_k(M, M')$, then by Lemma 4.3.1, l_k prefers some $s_r \in M(l_k) \setminus M'(l_k)$ to s; and by Lemma 4.3.4, l_k prefers s to some $s_z \in M'(l_k) \setminus M(l_k)$. Consequently, l_k prefers a student in $M(l_k) \setminus M'(l_k)$ to one in $M'(l_k) \setminus M(l_k)$, contradicting $M' \preceq_L M$. If instead $s \in M'(l_k) \setminus M(l_k)$, then by Lemma 4.3.3, l_k prefers M to M', again a contradiction. Hence no student prefers M' to M. Therefore each student is either indifferent between M and M' or prefers M to M', i.e. $M \preceq_S M'$. □

4.4 Stable matchings in SPA-S form a distributive lattice

To show that $(\mathcal{M}, \preceq)$ forms a distributive lattice, we define the *meet* and *join* of any two stable matchings in $\mathcal{M}$ based on student preferences. Given two stable matchings M and M', the meet matching assigns each student to the project they prefer more between their projects in M and M', while the join matching assigns each student to the less preferred of the two. In Lemmas 4.4.4 and 4.4.8, we show that both the meet and join matchings are stable. These results show that the meet and join operations are well-defined in $\mathcal{M}$ and respect the dominance relation $\preceq$. Finally, in Theorem 4.4.1, we prove that the meet and join operations distribute, and that $(\mathcal{M}, \preceq)$ is a distributive lattice.

Definition 4.4.1. Let M and M' be two stable matchings in I, and define a matching $M^\wedge$ as follows: for each student s_i,

- if s_i is unassigned in both M and M', then s_i is unassigned in $M^\wedge$;
- if s_i is assigned to the same project in both M and M', then s_i is assigned to that project in $M^\wedge$.
- otherwise, s_i is assigned in $M^\wedge$ to the better of their projects in M and M'.

In Lemma 4.4.4, we prove that $M^\wedge$ is a stable matching in I. To show this, we present Lemmas 4.4.1 – 4.4.3.

Lemma 4.4.1. *If a lecturer l_k is undersubscribed in $M^\wedge$, then l_k is undersubscribed in both M and M'.*

Proof. Suppose, for contradiction, that l_k is undersubscribed in $M^\wedge$, but is full in both M and M'. Then, $|M(l_k)| > |M^\wedge(l_k)|$ and $|M'(l_k)| > |M^\wedge(l_k)|$. Thus, there exists projects $p_a, p_b \in P_k$ such that

$$|M(p_a)| > |M^\wedge(p_a)| \quad \text{and} \quad |M'(p_b)| > |M^\wedge(p_b)|.$$

Suppose that $s_a \in M(p_a) \setminus M^\wedge(p_a)$ and $s_b \in M'(p_b) \setminus M^\wedge(p_b)$. This implies that $s_a \in M(p_a) \setminus M'(p_a)$ and $s_b \in M'(p_b) \setminus M(p_b)$. By the construction of $M^\wedge$, each student is assigned to the more preferred of their two projects in M and M'. Therefore, (a) s_a prefers $M'(s_a)$ to p_a, and (b) s_b prefers $M(s_b)$ to p_b. We claim that p_a is undersubscribed in M' and p_b is undersubscribed in M. We now consider each of these cases in turn.

Case (a): $s_a \in M(p_a) \setminus M'(p_a)$ and s_a prefers M' to M. Suppose for contradiction that p_a is full in M'. Since p_a cannot be oversubscribed in M, we have $|M'(p_a)| \geq |M(p_a)|$. Given that $|M(p_a)| > |M^\wedge(p_a)|$, it follows that $|M'(p_a)| > |M^\wedge(p_a)|$. Hence there exists some student $s \in M'(p_a) \setminus M^\wedge(p_a)$, which means $s \in M'(p_a) \setminus M(p_a)$. Moreover, by the construction of $M^\wedge$, s

prefers M to M'. Applying the first part of Lemma 4.3.3 to the matchings M and M', with s_a as a student who prefers M' to M and $s \in M'(p_a) \setminus M(p_a)$, it follows that l_k prefers s_a to s (note that here, M and M' are swapped compared to Lemma 4.3.3). On the other hand, applying the same lemma to M and M', with s as a student who prefers M to M' and $s_a \in M(p_a) \setminus M'(p_a)$, it follows that l_k prefers s to s_a. This yields a contradiction. Therefore, p_a is undersubscribed in M'.

Case (b): Suppose $s_b \in M'(p_b) \setminus M(p_b)$ and s_b prefers M to M'. Following a similar argument to case (a), if, on the contrary, p_b were full in M, then $|M(p_b)| > |M^\wedge(p_b)|$, and there would exist some student $s \in M(p_b) \setminus M^\wedge(p_b)$, and hence $s \in M(p_b) \setminus M'(p_b)$. By the construction of $M^\wedge$, s prefers M' to M. In this case, we have $s_b \in M'(p_b) \setminus M(p_b)$ who prefers M to M', and $s \in M(p_b) \setminus M'(p_b)$ who prefers M' to M. Applying Lemma 4.3.3 to these two cases yields a contradiction on l_k's preference list. Hence, p_b is undersubscribed in M.

From cases (a) and (b), it follows that s_a prefers $M'(s_a)$ to p_a, where p_a is undersubscribed in M', and that s_b prefers $M(s_b)$ to p_b, where p_b is undersubscribed in M. By the second part of Lemma 4.3.3, this implies that l_k prefers s_a to each student in $M'(l_k) \setminus M(l_k)$, and also prefers s_b to each student in $M(l_k) \setminus M'(l_k)$. If $s_b \in M'(l_k) \setminus M(l_k)$, then l_k prefers s_a to s_b. If instead $s_b \in S_k(M, M')$, then since s_b prefers M to M', Lemma 4.3.1 implies that there exists some $s' \in M'(l_k) \setminus M(l_k)$ where l_k prefers s' to s_b. Thus, l_k prefers s_a to s', and hence l_k prefers s_a to s_b. On the other hand, since s_b prefers M to M' and p_b is undersubscribed in M, then Lemma 4.3.3 implies that l_k prefers s_b to each student in $M(l_k) \setminus M'(l_k)$. If $s_a \in M(l_k) \setminus M'(l_k)$, then l_k prefers s_b to s_a, a contradiction. If instead $s_a \in S_k(M, M')$, then, since s_a prefers M' to M, there exists some $s \in M(l_k) \setminus M'(l_k)$ such that l_k prefers s to s_a. Consequently, l_k prefers s_b to s, and therefore to s_a. This again yields a contradiction.

Therefore, our assumption is false, and l_k is undersubscribed in both M and M'. □

Lemma 4.4.2. *If a project p_j is undersubscribed in $M^\wedge$, then it is undersubscribed in at least one of M or M'.*

Proof. Let l_k be the lecturer who offers project p_j. Suppose, for contradiction, that p_j is full in both M and M', but undersubscribed in $M^\wedge$. Then $|M(p_j)| > |M^\wedge(p_j)|$ and $|M'(p_j)| > |M^\wedge(p_j)|$. It follows that there exists a student $s_a \in M(p_j) \setminus M^\wedge(p_j)$. Since any student assigned to p_j in both M and M' must also be assigned to p_j in $M^\wedge$, we conclude that $s_a \notin M'(p_j)$, and hence $s_a \in M(p_j) \setminus M'(p_j)$. Similarly, there exists a student $s_b \in M'(p_j) \setminus M^\wedge(p_j)$, which implies that $s_b \in M'(p_j) \setminus M(p_j)$. By the construction of $M^\wedge$, s_a prefers M' to M, and s_b prefers M to M'.

By applying the first part of Lemma 4.3.3 to the matchings M' and M, with s_a as a student who prefers M' to M, and with $s_b \in M'(p_j) \setminus M(p_j)$, it follows that l_k prefers s_a to s_b (note that here, M and M' are swapped compared to Lemma 4.3.3). Conversely, applying the same lemma to M and M', with s_b as a student who prefers M to M' and $s_a \in M(p_j) \setminus M'(p_j)$, we conclude that l_k

prefers s_b to s_a. This is a contradiction, since l_k cannot simultaneously prefer s_a to s_b and s_b to s_a. Therefore, p_j must be undersubscribed in at least one of M or M'. □

Lemma 4.4.3. *$M^\wedge$ is a matching.*

Proof. By construction, no student is assigned to more than one project in $M^\wedge$. It remains to show that no project or lecturer is oversubscribed in $M^\wedge$. Suppose, for contradiction, that some project p_j is oversubscribed in $M^\wedge$. Let l_k be the lecturer who offers p_j. Then $|M^\wedge(p_j)| > |M(p_j)|$ and $|M^\wedge(p_j)| > |M'(p_j)|$, since both M and M' are valid matchings. Thus, there exist students $s_a \in M^\wedge(p_j) \setminus M'(p_j)$ and $s_b \in M^\wedge(p_j) \setminus M(p_j)$. It follows that $s_a \in M(p_j) \setminus M'(p_j)$ and $s_b \in M'(p_j) \setminus M(p_j)$, where s_a prefers M to M' and s_b prefers M' to M.

By stability of M' and since s_a prefers p_j to $M'(s_a)$, it follows that l_k prefers the worst student in $M'(p_j)$ to s_a (if p_j is full in M') or the worst student in $M'(l_k)$ to s_a (if p_j is undersubscribed in M'). This implies that l_k prefers s_b to s_a, since $s_b \in M'(p_j)$. On the other hand, s_b prefers p_j to $M(s_b)$. If p_j is full in M, then l_k prefers s_b to some student in $M(p_j)$ (namely s_a), and (s_b, p_j) blocks M, a contradiction. If p_j is undersubscribed in M, then l_k prefers s_b to some student in $M(l_k)$ (namely s_a), and (s_b, p_j) blocks M, a contradiction (This holds whether l_k is full or undersubscribed in M). Therefore, our assumption is false and no project is oversubscribed in $M^\wedge$.

Next, suppose for contradiction that some lecturer l_k is oversubscribed in $M^\wedge$. Then there exist projects p_a and p_b offered by l_k such that $|M^\wedge(p_a)| > |M'(p_a)|$ and $|M^\wedge(p_b)| > |M(p_b)|$. Since both M and M' are valid matchings, this implies that p_a is undersubscribed in M' and p_b is undersubscribed in M. Moreover, as established earlier, no project is oversubscribed in $M^\wedge$. Let $s_a \in M^\wedge(p_a) \setminus M'(p_a)$, so $s_a \in M(p_a) \setminus M'(p_a)$, and let $s_b \in M^\wedge(p_b) \setminus M(p_b)$, so $s_b \in M'(p_b) \setminus M(p_b)$. By the definition of $M^\wedge$, each student is assigned to the more preferred of their two projects in M and M'; therefore, s_a prefers M to M', and s_b prefers M' to M.

Since p_a is undersubscribed in M', we have $s_a \notin M'(l_k)$; otherwise (s_a, p_a) would block M', regardless of whether l_k is full or undersubscribed in M'. Similarly, since p_b is undersubscribed in M, we have $s_b \notin M(l_k)$, otherwise (s_b, p_b) would block M. Hence $s_a \in M(l_k) \setminus M'(l_k)$ and $s_b \in M'(l_k) \setminus M(l_k)$. Now, by Lemma 4.3.2, since s_a prefers M to M', it follows that l_k prefers M' to M. Conversely, since s_b prefers M' to M, the same lemma implies that l_k prefers M to M'. This yields a contradiction. Hence our assumption is false, and no lecturer is oversubscribed in $M^\wedge$. Therefore, $M^\wedge$ is a valid matching. □

Lemma 4.4.4. *$M^\wedge$ is a stable matching.*

Proof. Suppose for contradiction that (s, p) is a blocking pair for $M^\wedge$, where project p is offered by lecturer l. Then either:

(S1) s is unassigned in $M^\wedge$, or

(S2) s is assigned in $M^\wedge$, but prefers p to $M^\wedge(s)$.

And one of the following four conditions holds for p and l:

(P1) both p and l are undersubscribed in $M^\wedge$.

(P2) p is undersubscribed in $M^\wedge$, l is full in $M^\wedge$, and $s \in M^\wedge(l)$.

(P3) p is undersubscribed in $M^\wedge$, l is full in $M^\wedge$, and l prefers s to the worst student in $M^\wedge(l)$.

(P4) p is full in $M^\wedge$, and l prefers s to the worst student in $M^\wedge(p)$.

We consider each combination of conditions in turn. Note that the case (S1 & P2) cannot arise, since s is unassigned in $M^\wedge$ and therefore cannot belong to $M^\wedge(l)$.

(S1 & P1) and (S2 & P1): First, suppose s is unassigned in $M^\wedge$. Then, by the construction of $M^\wedge$, s is unassigned in both M and M'. Alternatively, if s is assigned in $M^\wedge$ and prefers p to $M^\wedge(s)$, then s prefers p to both $M(s)$ and $M'(s)$, since s receives their preferred project in $M^\wedge$. Now consider condition (P1), where both p and its lecturer l are undersubscribed in $M^\wedge$. By Lemma 4.4.1, since l is undersubscribed in $M^\wedge$, it is undersubscribed in both M and M'. By Lemma 4.4.2, since p is undersubscribed in $M^\wedge$, it is undersubscribed in at least one of M or M'. Without loss of generality, suppose p is undersubscribed in M'. Then, whether s is unassigned in M', or assigned to a project they prefer less than p, the pair (s, p) blocks M', a contradiction. We conclude that (S1 & P1) and (S2 & P1) cannot arise.

(S2 & P2), (S1 & P3) and (S2 & P3): First, consider (S1), where s is unassigned in $M^\wedge$. By construction of $M^\wedge$, this means that s is unassigned in both M and M'. Next, consider (S2), where s is assigned in $M^\wedge$ and prefers p to $M^\wedge(s)$. Since each student in $M^\wedge$ receives the more preferred of their two projects from M and M', it follows that s prefers p to both $M(s)$ and $M'(s)$. In conditions (P2) and (P3), p is undersubscribed in $M^\wedge$. Hence, by Lemma 4.4.2, p must be undersubscribed in at least one of M or M'.

Suppose first that p is undersubscribed in both M and M'. From (S2 & P2), we have that $s \in M^\wedge(l)$, which means that $s \in M(l)$ or $s \in M'(l)$. If $s \in M'(l)$, then (s, p) blocks M', since s prefers p to $M'(s)$ and p is undersubscribed in M'. This blocking pair arises whether l is undersubscribed or full in M'. A similar contradiction arises in M if $s \in M(l)$. In conditions (S1 & P3) and (S2 & P3), let s_z be the worst student in $M^\wedge(l)$, where l prefers s to s_z. If $s_z \in M'(l)$, then (s, p) blocks M', since s is either unassigned in M' or prefers p to $M'(s)$, p is undersubscribed in M' and l prefers s to s_z (again, this holds whether l is full or undersubscribed in M'). A similar contradiction arises in M if $s_z \in M(l)$.

Next, suppose without loss of generality that p is full in M but undersubscribed in $M^\wedge$ and M'. If l is undersubscribed in M', then the pair (s, p) blocks M', since s is either unassigned in M' or prefers p to $M'(s)$, and both p and l are undersubscribed in M'. Hence l is full in M', and therefore also full in M. From (S2 & P2), we have that $s \in M(l)$ or $s \in M'(l)$. If $s \in M'(l)$,

then (s, p) blocks M', since s prefers p to $M'(s)$, p is undersubscribed in M' and l is full in M'. Therefore, $s \in M(l) \setminus M'(l)$. In (S1 & P3) and (S2 & P3), we have that $s_z \in M(l)$ or $s_z \in M'(l)$. If $s_z \in M'(l)$, then (s, p) blocks M', since s is either unassigned in M' or prefers p to $M'(s)$, p is undersubscribed in M', l is full in M', and l prefers s to s_z. Thus $s_z \in M(l) \setminus M'(l)$.

Now since p is full in M but undersubscribed in $M^\wedge$, it follows that $|M(p)| > |M^\wedge(p)|$. Since l is full in both $M^\wedge$ and M, and $|M(p)| > |M^\wedge(p)|$, there exists a project p_a offered by l such that $|M^\wedge(p_a)| > |M(p_a)|$, implying that p_a is undersubscribed in M. Thus, there exists a student $s_a \in M^\wedge(p_a) \setminus M(p_a)$, and hence $s_a \in M'(p_a) \setminus M(p_a)$. Since s_a receives their more preferred project in $M^\wedge$, they prefer p_a to $M(s_a)$. Moreover by the stability of M (and since p_a is undersubscribed in M), l prefers the worst student in $M(l)$ to s_a. In the (S2 & P2) case, it follows that l prefers s to s_a, since $s \in M(l) \setminus M'(l)$. However, since s prefers p to $M'(s)$, p is undersubcribed in M', l is full in M', and l prefers s to s_a, the pair (s, p) blocks M', a contradiction. In the (S1 & P3) and (S2 & P3) case, it follows that l prefers s_z to s_a, since $s_z \in M(l) \setminus M'(l)$. Consequently, l prefers s to s_a, since l prefers s to s_z. Again, we arrive at a similar contradiction as in (S2 & P2) case, whereby (s, p) blocks M'.

Therefore, no blocking pair of type (S2 & P2), (S1 & P3) and (S2 & P3) exists in $M^\wedge$.

(S1 & P4) and (S2 & P4): Clearly, if s is unassigned in $M^\wedge$, then by construction, s is unassigned in both M and M'. If instead s is assigned in $M^\wedge$ and prefers p to $M^\wedge(s)$, then s prefers p to both $M(s)$ and $M'(s)$, since s receives the more preferred of their two projects in $M^\wedge$. Consider condition (P4), where p is full in $M^\wedge$ and l prefers s to the worst student in $M^\wedge(p)$. Let s_z be the worst student in $M^\wedge(p)$. Clearly, either $s_z \in M(p)$ or $s_z \in M'(p)$.

First suppose $(s_z, p) \in M$. If s is unassigned in M, or if s prefers p to $M(s)$, then (s, p) blocks M, since p is full and l prefers s to $s_z \in M(p)$. This contradicts the stability of M. A similar argument applies if $(s_z, p) \in M'$: whether s is unassigned in M', or prefers p to $M'(s)$, the pair (s, p) blocks M', again a contradiction. Therefore, no blocking pair of type (S1 & P4) or (S2 & P4) can exist in $M^\wedge$.

In all possible cases, we arrive at a contradiction. Therefore, no blocking pair exists in $M^\wedge$, and $M^\wedge$ is stable. □

We denote by $M \wedge M'$ the set of (student, project) pairs in which each student is assigned the better of her project in M and M'; and it follows from Lemma 4.4.4 that $M \wedge M'$ is a stable matching. Hence, if each student is given the better of her project in any fixed set of stable matchings, then the resulting assignment is a stable matching. For the case where $\mathcal{M}$ is the set of all stable matchings in I, we denote by $\bigwedge_{M \in \mathcal{M}} M$, or simply $\bigwedge \mathcal{M}$, the resulting stable matching. This matching is student-optimal and, by Corollary 4.3.1, lecturer-pessimal.

Definition 4.4.2. Let M and M' be two stable matchings in I, and define a matching $M^\vee$ as follows: for each student s_i,

- if s_i is unassigned in both M and M', then s_i is unassigned in $M^\vee$;
- if s_i is assigned to the same project in both M and M', then s_i is assigned to that project in $M^\vee$;
- otherwise, s_i is assigned in $M^\vee$ to the worse of their two projects in M and M'.

In Lemma 4.4.8, we prove that $M^\vee$ is a stable matching in I. To prove this, we first present Lemmas 4.4.5 – 4.4.7.

Lemma 4.4.5. *If a lecturer l_k is undersubscribed in $M^\vee$, then l_k is undersubscribed in both M and M'.*

Proof. Suppose, for contradiction, that l_k is undersubscribed in $M^\vee$, but is full in both M and M'. Then $|M(l_k)| > |M^\vee(l_k)|$ and $|M'(l_k)| > |M^\vee(l_k)|$. It follows that there exists student-project pairs (s_a, p_a) and (s_b, p_b) such that $(s_a, p_a) \in M \setminus M'$ and $(s_b, p_b) \in M' \setminus M$ (since $M^\vee(l_k) \neq M(l_k)$ and $M^\vee(l_k) \neq M'(l_k)$). By construction of $M^\vee$, each student is assigned to the less preferred of their two projects in M and M'; therefore, s_a prefers M to M', and s_b prefers M' to M.

By Lemma 4.3.2, since s_a prefers M to M', it follows that l_k prefers M' to M. Conversely, since s_b prefers M' to M, the same lemma implies that l_k prefers M to M'. This yields a contradiction. Therefore, our assumption is false, and l_k must be undersubscribed in both M and M'. □

Lemma 4.4.6. *If a project p_j is undersubscribed in $M^\vee$, then it must be undersubscribed in at least one of M or M'.*

Proof. Suppose, for contradiction, that p_j is undersubscribed in $M^\vee$, but full in both M and M'. Then we have

$$|M(p_j)| > |M^\vee(p_j)| \quad \text{and} \quad |M'(p_j)| > |M^\vee(p_j)|.$$

It follows that there exist students $s_a \in M(p_j) \setminus M^\vee(p_j)$ and $s_b \in M'(p_j) \setminus M^\vee(p_j)$. In particular, $s_a \in M(p_j) \setminus M'(p_j)$ and $s_b \in M'(p_j) \setminus M(p_j)$. Since each student is assigned in $M^\vee$ to the less preferred of their projects in M and M', it follows that s_a prefers M to M', and s_b prefers M' to M.

Let l_k be the lecturer who offers p_j. By stability of M', since s_a prefers p_j to $M'(s_a)$ and p_j is full in M', it follows that l_k prefers the worst student in $M'(p_j)$ to s_a. In particular, l_k prefers student $s_b \in M'(p_j)$ to s_a. However, since s_b prefers p_j to $M(s_b)$, p_j is full in M, and l_k prefers s_b to some student in $M(p_j)$ (namely s_a), it follows that (s_b, p_j) blocks M, a contradiction. Therefore, p_j must be undersubscribed in at least one of M or M'. □

Lemma 4.4.7. *$M^\vee$ is a matching.*

Proof. By construction, no student is assigned to more than one project in $M^\vee$. It remains to show that no project or lecturer is oversubscribed in $M^\vee$. Suppose, for contradiction, that some project p_j is oversubscribed in $M^\vee$. Then

$$|M^\vee(p_j)| > |M(p_j)| \quad \text{and} \quad |M^\vee(p_j)| > |M'(p_j)|.$$

Thus, there exist students $s_a \in M^\vee(p_j) \setminus M'(p_j)$ and $s_b \in M^\vee(p_j) \setminus M(p_j)$. Since any student assigned to p_j in both M and M' would also be assigned to p_j in $M^\vee$, it follows that $s_a \in M(p_j) \setminus M'(p_j)$ and $s_b \in M'(p_j) \setminus M(p_j)$. Moreover, s_a prefers M' to M, and s_b prefers M to M', since each student is assigned in $M^\vee$ to the less preferred of their two projects. Let l_k be the lecturer who offers p_j. By the first part of Lemma 4.3.3, since s_a prefers M' to M and $s_b \in M'(p_j) \setminus M(p_j)$, it follows that l_k prefers s_a to s_b. Similarly, since s_b prefers M to M' and $s_a \in M(p_j) \setminus M'(p_j)$, it follows that l_k prefers s_b to s_a. Thus, l_k simultaneously prefers s_a to s_b, and s_b to s_a, a contradiction. Therefore, no project is oversubscribed in $M^\vee$.

Next, suppose that there exists some lecturer l_k who is oversubscribed in $M^\vee$. Then there must be some project $p_a \in P_k$ such that $|M^\vee(p_a)| > |M'(p_a)|$, meaning that p_a is undersubscribed in M', since no project can be oversubscribed in $M^\vee$. Similarly, there exists some project $p_b \in P_k$ such that $|M^\vee(p_b)| > |M(p_b)|$, meaning that p_b is undersubscribed in M. Let s_a be a student such that $(s_a, p_a) \in M^\vee \setminus M'$. Thus, $s_a \in M(p_a) \setminus M'(p_a)$ and s_a prefers M' to M. Let s_b be a student such that $(s_b, p_b) \in M^\vee \setminus M$; thus $s_b \in M'(p_b) \setminus M(p_b)$ and s_b prefers M to M'.

By Lemma 4.3.3, since s_a prefers M' to M and p_a is undersubscribed in M', l_k prefers s_a to each student in $M'(l_k) \setminus M(l_k)$ (Note that here, M and M' are swapped compared to the statement of Lemma 4.3.3). If $s_b \in M'(l_k) \setminus M(l_k)$, then l_k prefers s_a to s_b. If instead $s_b \in S_k(M, M')$, then since s_b prefers M to M', Lemma 4.3.1 implies that there exists some $s' \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s' to s_b. It follows that l_k prefers s_a to s', and consequently to s_b.

On the other hand, by Lemma 4.3.3 again, since s_b prefers M to M' and p_b is undersubscribed in M, it follows that l_k prefers s_b to each student in $M(l_k) \setminus M'(l_k)$. If $s_a \in M(l_k) \setminus M'(l_k)$, then l_k prefers s_b to s_a, a contradiction. If instead $s_a \in S_k(M, M')$, then by Lemma 4.3.1, there exists some $s \in M(l_k) \setminus M'(l_k)$ where l_k prefers s to s_a (Note that here s_a prefers M' to M). This implies that l_k prefers s_b to s, and consequently to s_a. This yields a contradiction on l_k's preferences. Hence, no lecturer is oversubscribed in $M^\vee$. Therefore, $M^\vee$ is a matching. □

Lemma 4.4.8. *$M^\vee$ is a stable matching.*

Proof. Suppose for contradiction that (s, p) is a blocking pair for $M^\vee$, where project p is offered by lecturer l. Then either:

(S1) s is unassigned in $M^\vee$, or

(S2) s is assigned in $M^\vee$, but prefers p to $M^\vee(s)$.

And one of the following four conditions holds for the p and l:

(P1) both p and l are undersubscribed in $M^\vee$;

(P2) p is undersubscribed in $M^\vee$, l is full in $M^\vee$, and $s \in M^\vee(l)$;

(P3) p is undersubscribed in $M^\vee$, l is full in $M^\vee$, and l prefers s to the worst student in $M^\vee(l)$;

(P4) p is full in $M^\vee$, and l prefers s to the worst student in $M^\vee(p)$.

We consider each combination of conditions in turn. Note that the case (S1 & P2) cannot arise, since s is unassigned in $M^\vee$ and therefore cannot belong to $M^\vee(l)$.

(S1 & P1): Suppose s is unassigned in $M^\vee$, and both p and l are undersubscribed in $M^\vee$. By construction, if s is unassigned in $M^\vee$, then s is unassigned in both M and M'. By Lemma 4.4.5, since l is undersubscribed in $M^\vee$, it is undersubscribed in both M and M'. By Lemma 4.4.6, since p is undersubscribed in $M^\vee$, it is undersubscribed in at least one of M or M'. Without loss of generality, suppose p is undersubscribed in M'. Then s is unassigned in M', and both p and l are undersubscribed in M', the pair (s, p) blocks M', a contradiction. A similar argument applies if p is undersubscribed in M. We conclude that no blocking pair of type (S1 & P1) exists in $M^\vee$.

(S2 & P1): Suppose s is assigned in $M^\vee$, prefers p to $M^\vee(s)$, and both p and l are undersubscribed in $M^\vee$. Since l is undersubscribed in $M^\vee$, it follows from Lemma 4.4.5 that l is undersubscribed in both M and M'. In addition, Lemma 4.4.6 implies that p is undersubscribed in at least one of M or M'. Without loss of generality, assume that p is undersubscribed in M'. Let $p^* = M^\vee(s)$. Since s is assigned in $M^\vee$, they must be assigned in at least one of M or M', and p^* is the less preferred of the two. There are two possibilities for $M(s)$ and $M'(s)$:

(i) $M'(s) = p^*$. Then s prefers p to $M'(s)$. Since s prefers p to p^*, and both p and l are undersubscribed in M', the pair (s, p) blocks M', a contradiction.

(ii) $M(s) = p^*$. Then $s \in M(p^*) \setminus M'(p^*)$. Suppose first that p is undersubscribed in M. Then, since s prefers p to p^*, and both p and l are undersubscribed in M, the pair (s, p) blocks M, a contradiction. Next suppose that p is full in M. Since p is undersubscribed in both $M^\vee$ and M', it follows that

$$|M(p)| > |M^\vee(p)| \quad \text{and} \quad |M(p)| > |M'(p)|.$$

Hence, there exists some student $s' \in M(p) \setminus M^\vee(p)$, and in particular $s' \in M(p) \setminus M'(p)$. Since s' is assigned in $M^\vee$ to the less preferred of their two projects from M and M', it follows that s' prefers M to M'. Since both p and l are undersubscribed in M', the pair (s', p) blocks M', contradicting the stability of M'.

In both cases (i) and (ii), a contradiction arises. Therefore, no blocking pair of type (S2 & P1) exists in $M^\vee$.

(S2 & P2): Here, s is assigned in $M^\vee$, prefers p to $M^\vee(s)$, p is undersubscribed in $M^\vee$, l is full in $M^\vee$, and $s \in M^\vee(l)$. By Lemma 4.4.6, p is undersubscribed in at least one of M or M'; without loss of generality, assume p is undersubscribed in M'. Let $p^* = M^\vee(s)$, where p^* is offered by l. Since s is assigned in $M^\vee$, they must be assigned in either M or M' (or both). We consider the possibilities for $M(s)$ and $M'(s)$:

(i) $M'(s) = p^*$. Then s prefers p to $M'(s)$. Moreover, it follows that $s \in M'(l)$, and since p is undersubscribed in M', the pair (s, p) blocks M', a contradiction (This blocking pair occurs whether l is undersubscribed or full in M').

(ii) $M(s) = p^*$ and $M'(s) \neq p^*$. Then $s \in M(p^*) \setminus M'(p^*)$ and prefers p to $M(s)$. If p is undersubscribed in M, then since $s \in M(l)$, the pair (s, p) blocks M. Therefore, p must be full in M. Following the case (ii) argument in (S2 & P1) (where p is full in M but undersubscribed in both $M^\vee$ and M'), there exists some $s^* \in M(p) \setminus M'(p)$ who prefers p to $M'(s^*)$. We now consider the following subcases:

(iia): $s \in M'(l)$. Recall that s prefers p to $M(s)$ and p is full in M. By the stability of M, l prefers the worst student in $M(p)$ to s; that is, l prefers s^* to s. Now since s^* prefers p to $M'(s)$, p is undersubscribed in M' and l prefers s^* to some student in $M'(l)$ (namely s), the pair (s^*, p) blocks M', a contradiction. (Again, this blocking pair occurs whether l is undersubscribed or full in M')

(iib:) Suppose $s \notin M'(l)$. Then $s \in M(l) \setminus M'(l)$, so there exists some $\hat{s} \in M'(l) \setminus M(l)$, since $|M(l)| = |M'(l)|$. Since $s \in M^\vee(l)$ and $s \in M(l)$, it follows that s prefers M' to M, i.e. s prefers $M'(s)$ to p^*. Since p^* is undersubscribed in M' and $\hat{s} \in M'(l) \setminus M(l)$, Lemma 4.3.3 implies that l prefers s to $\hat{s}$. Now, recall that s prefers p to $M(s)$ and p is full in M. By the stability of M, l prefers the worst student in $M(p)$ to s; that is, l prefers s^* to s. Since, l prefers s^* to s, it follows that l prefers s^* to $\hat{s}$. However, s^* prefers p to $M'(s^*)$, p is undersubscribed in M', and l prefers s^* to some student in $M'(l)$ (namely $\hat{s}$). Thus (s^*, p) blocks M', a contradiction.

Therefore, no blocking pair of type (S2 & P2) exists in $M^\vee$.

(S1 & P3): Suppose s is unassigned in $M^\vee$, p is undersubscribed in $M^\vee$, l is full in $M^\vee$, and l prefers s to the worst student in $M^\vee(l)$. By construction of $M^\vee$, any student unassigned in $M^\vee$ must also be unassigned in both M and M'. Let s_z denote the worst student in $M^\vee(l)$, so that l prefers s to s_z. Since p is undersubscribed in $M^\vee$, by Lemma 4.4.6, p is undersubscribed in at least one of M or M'; without loss of generality, assume that p is undersubscribed in M' (it may

be full or undersubscribed in M). Suppose first that $s_z \in M'(l)$. Then in M', s is unassigned, p is undersubscribed, and l prefers s to s_z. Therefore, the pair (s, p) blocks M', contradicting its stability. It follows that $s_z \notin M'(l)$, and hence $s_z \in M(l) \setminus M'(l)$.

Since $s_z \in M^\vee(l)$, they must be assigned in $M^\vee$ to the less preferred of their two projects. Hence s_z prefers M' to M. Let $M(s_z) = p_z$. If p_z is full in M', then $|M'(p_z)| \geq |M(p_z)|$. Since $s_z \in M(p_z) \setminus M'(p_z)$, there exists some $s^* \in M'(p_z) \setminus M(p_z)$. Since s_z prefers M' to M, Lemma 4.3.3 implies that l prefers s_z to s^*. Similarly, given that $s_z \in M(l) \setminus M'(l)$, there also exist some student $s^* \in M'(l) \setminus M(l)$. Now if p_z is undersubscribed in M', then by the second part of Lemma 4.3.3, l prefers s_z to s^*. In both cases, l prefers s_z to s^*. Since l also prefers s to s_z, it follows that l prefers s to s^*. But in M', s is unassigned, p is undersubscribed, and l prefers s to some $s^* \in M'(l)$. Hence (s, p) blocks M', a contradiction. Note that the pair (s, p) blocks M' whether l is undersubscribed or full in M'.

It follows that no blocking pair of type (S1 & P3) exists in $M^\vee$.

(S2 & P3): Suppose s is assigned in $M^\vee$, prefers p to $M^\vee(s)$, p is undersubscribed in $M^\vee$, l is full in $M^\vee$, and l prefers s to the worst student in $M^\vee(l)$. Let $p^* = M^\vee(s)$, and suppose that $p^* = M'(s)$, so s prefers p to p^*. Let s_z be the worst student in $M^\vee(l)$. By Lemma 4.4.6, p is undersubscribed in at least one of M or M'.

Suppose first that p is undersubscribed in M' (it may be full or undersubscribed in M). If $s_z \in M'(l)$, then in M', s is assigned to p^*, prefers p to p^*, p is undersubscribed in M', and l prefers s to s_z. Thus, the pair (s, p) blocks M', a contradiction. It follows that $s_z \in M(l) \setminus M'(l)$. Since $s_z \in M^\vee(l)$, they are assigned in both M and M', and assigned in $M^\vee$ to the less preferred of the two. Therefore, s_z prefers M' to M. Let $M(s_z) = p_z$, where p_z is offered by l. Regardless of whether p_z is full or undersubscribed in M', by Lemma 4.3.3, there exists some student $s^* \in M'(l)$ such that l prefers s_z to s^*. Since l prefers s to s_z, it follows that l also prefers s to s^*. However, since s prefers p to $M'(s)$, and p is undersubscribed in M', the pair (s, p) blocks M', a contradiction.

Now suppose instead that p is full in M' and undersubscribed in M. Then $|M'(p)| > |M^\vee(p)|$ and $|M'(p)| > |M(p)|$, so there exists a student $s^* \in M'(p) \setminus M^\vee(p)$, and in particular $s^* \in M'(p) \setminus M(p)$. Moreover, s^* prefers p to $M(s)$. By the stability of M and since p is undersubscribed in M, it follows that $s^* \notin M(l)$, and l prefers each student in $M(l)$ to s^*. Thus, $s^* \in M'(l) \setminus M(l)$. If $s_z \in M(l)$, then l prefers s_z to s^*, and since l also prefers s to s_z, it follows that l prefers s to s^*. Hence, the pair (s, p) blocks M', a contradiction. We conclude that $s_z \in M'(l) \setminus M(l)$. Since $s_z \in M^\vee(l)$, it follows that s_z prefers M to M'. Let $M'(s_z) = p_z$, where p_z is offered by l.

Suppose p_z is full in M. Then there exists a student $\hat{s} \in M(p_z) \setminus M'(p_z)$. (This is because $s_z \in M'(p_z) \setminus M(p_z)$ and clearly $|M(p_z)| \geq |M'(p_z)|$). Since s_z prefers M to M', Lemma 4.3.3 implies that l prefers s_z to $\hat{s}$. Moreover, since $\hat{s} \in M(l)$, and l prefers each student in $M(l)$ to s^*, it follows that l prefers $\hat{s}$ to s^*, and l prefers s to s^* (since l prefers s to s_z and l prefers s_z to $\hat{s}$). Suppose p_z is undersubscribed in M. Then Lemma 4.3.3 implies that there exists a student

$\hat{s} \in M(l) \setminus M'(l)$ such that l prefers s_z to $\hat{s}$. Since $\hat{s} \in M(l)$, and l prefers each student in $M(l)$ to s^*, it follows that l prefers $\hat{s}$ to s^*, and consequently, prefers s to s^*(since l prefers s to s_z and l prefers s_z to $\hat{s}$). In both cases, l prefers s to some student $s^* \in M'(p)$. Since s prefers p to $M'(s)$, and p is full in M', the pair (s, p) blocks M', a contradiction.

A similar argument applies if $p^* = M(s)$. We therefore conclude that no blocking pair of type (S2 & P3) exists in $M^\vee$.

(S1 & P4): Suppose s is unassigned in $M^\vee$, project p is full in $M^\vee$, and lecturer l prefers s to the worst student in $M^\vee(p)$. Then, by construction of $M^\vee$, student s is unassigned in both M and M'. Let s_z denote the worst student in $M^\vee(p)$, so that l prefers s to s_z. Since $s_z \in M^\vee(p)$, it must be that either $(s_z, p) \in M$ or $(s_z, p) \in M'$.

Suppose first that $(s_z, p) \in M$. If p is full in M, then s is unassigned, p is full, and l prefers s to $s_z \in M(p)$, so the pair (s, p) blocks M, contradicting its stability. If instead p is undersubscribed in M, then s is unassigned, p is undersubscribed, and l prefers s to $s_z \in M(l)$; thus, (s, p) again blocks M, a contradiction. Now suppose that $(s_z, p) \in M'$. The same reasoning applies: whether p is full or undersubscribed in M', student s is unassigned in M', $s_z \in M'(p)$ and $s_z \in M'(l)$, and l prefers s to s_z. Hence, the pair (s, p) blocks M', again a contradiction.

We conclude that no blocking pair of type (S1 & P4) exists in $M^\vee$.

(S2 & P4): Suppose s is assigned in $M^\vee$, prefers p to $M^\vee(s)$, p is full in $M^\vee$, and l prefers s to the worst student in $M^\vee(p)$. Let $p^* = M^\vee(s)$, and suppose $p^* = M'(s)$, so that s prefers p to p^*. Let s_z be the worst student in $M^\vee(p)$. If $(s_z, p) \in M'$, then s prefers p to $M'(s)$, p is full in M', and l prefers s to $s_z \in M'(p)$. Moreover, if p is undersubscribed in M', then l prefers s to student $s_z \in M'(l)$. Therefore, the pair (s, p) blocks M', a contradiction. It follows that $s_z \in M(p) \setminus M'(p)$. Since $s_z \in M^\vee(p)$, they are assigned in both M and M', and assigned in $M^\vee$ to the less preferred of the two. Hence, s_z prefers M' to M.

Suppose first that p is full in M'. Then there exists some student $s^* \in M'(p) \setminus M(p)$. Since s_z prefers M' to M, Lemma 4.3.3 implies that l prefers s_z to s^*. If instead p is undersubscribed in M', then by Lemma 4.3.3, l prefers s_z to some student in $s^* \in M'(l) \setminus M(l)$. Since l also prefers s to s_z, it follows that l prefers s to student $s^* \in M'(l)$, namely s^*. Therefore, regardless of whether p is full or undersubscribed in M', the pair (s, p) blocks M', a contradiction. A similar argument applies if $p^* = M(s)$. We therefore conclude that no blocking pair of type (S2 & P4) can exist in $M^\vee$.

In all possible cases, we derive a contradiction. Therefore, no blocking pair exists in $M^\vee$, and $M^\vee$ is stable. □

We denote by $M \vee M'$ the set of (student, project) pairs in which each student is assigned to the poorer of her projects in M and M'; and it follows from Lemma $4.4.8$ that if each student is given

the poorer of her projects in any fixed set of stable matchings, then the resulting assignment is a stable matching. For the case where $\mathcal{M}$ is the set of all stable matchings in I, we denote by $\bigvee_{M\in\mathcal{M}} M$, or simply $\bigvee \mathcal{M}$, the resulting stable matching. This matching is student-pessimal and, by Corollary 4.3.1, lecturer-optimal. We are now ready to present our main result.

Theorem 4.4.1 ([121]). *Let I be an instance of* SPA-S*, and let $\mathcal{M}$ be the set of stable matchings in I. Let $\preceq$ be the dominance partial order on $\mathcal{M}$ and let $M, M' \in \mathcal{M}$. Then $(\mathcal{M}, \preceq)$ is a distributive lattice, with $M \wedge M'$ representing the meet of M and M', and $M \vee M'$ the join of M and M'.*

Proof. Let M and M' be two stable matchings in $\mathcal{M}$. By Lemma 4.4.4, we have that $M \wedge M'$ is a stable matching; and by the definition of $M \wedge M'$, it follows that $M \wedge M' \preceq M$ and $M \wedge M' \preceq M'$. Further, if M^* is an arbitrary stable matching satisfying $M^* \preceq M$ and $M^* \preceq M'$, then each student must be assigned in M^* to a project that is at least as good as her assigned projects in each of M and M', so that $M^* \preceq M \wedge M'$. Thus $M \wedge M'$ is the meet of M and M'. Similarly, by Lemma 4.4.8, we have that $M \vee M'$ is a stable matching; and by the definition of $M \vee M'$, it follows that $M \preceq M \vee M'$ and $M' \preceq M \vee M'$. Following a similar argument as above, $M \vee M'$ is the join of M and M'. Hence, $(\mathcal{M}, \preceq)$ is a lattice.

Next, we show that the join and meet operation distribute over each other. Let M, M' and M'' be stable matchings in $\mathcal{M}$. First, let $X = M \vee (M' \wedge M'')$ and let $Y = (M \vee M') \wedge (M \vee M'')$; we need to show that $X = Y$. Let s_i be an arbitrary student. If s_i is unassigned in each of M, M' and M'', it is clear that s_i is unassigned in both X and Y. Now, suppose that s_i is assigned to some project in each of M, M' and M''. We consider the following cases.

(i) Suppose that $M(s_i) = M'(s_i) = M''(s_i)$, clearly $X(s_i) = Y(s_i)$.

(ii) Suppose that either (a) $M(s_i) = M'(s_i)$ and $M(s_i) \neq M''(s_i)$ or (b) $M(s_i) \neq M'(s_i)$ and $M(s_i) = M''(s_i)$ holds. Irrespective of how we express s_i's preference over $M(s_i), M'(s_i)$ and $M''(s_i)$ in cases (a) and (b), we have that s_i is assigned to $M(s_i)$ in both X and Y.

(iii) Suppose that $M'(s_i) = M''(s_i)$ and $M'(s_i) \neq M(s_i)$. If s_i prefers $M'(s_i)$ to $M(s_i)$ then s_i is assigned to $M(s_i)$ in both X and Y. Otherwise, if s_i prefers $M(s_i)$ to $M'(s_i)$ then s_i is assigned to $M'(s_i)$ in both X and Y.

(iv) Suppose that $M(s_i)$, $M'(s_i)$ and $M''(s_i)$ are distinct projects. There are six different ways to express s_i's preference over $M(s_i)$, $M'(s_i)$ and $M''(s_i)$. If s_i prefers $M(s_i)$ to $M'(s_i)$ to $M''(s_i)$, then s_i is assigned to $M'(s_i)$ in both X and Y. If s_i prefers $M(s_i)$ to $M''(s_i)$ to $M'(s_i)$, then s_i is assigned to $M''(s_i)$ in both X and Y. We leave it to the reader to verify that in the remaining four cases, s_i is assigned to $M(s_i)$ in both X and Y.

Since s_i is an arbitrary student, it follows that $X = Y$; and thus the first distributive property holds. Next, we show that the second distributive property holds. Let $X = M \wedge (M' \vee M'')$ and let $Y = (M \wedge M') \vee (M \wedge M'')$. Let s_i be an arbitrary student. Again, if s_i is unassigned in each of M, M' and M'', it is clear that s_i is unassigned in both X and Y. Now, suppose s_i is assigned to some project in each of M, M' and M''. Following the same case analysis as before, we arrive at the same conclusion in cases (i) and (ii). We consider cases (iii) and (iv) in detail:

(iii) If $M'(s_i) = M''(s_i)$ and $M'(s_i) \neq M(s_i)$. If s_i prefers $M'(s_i)$ to $M(s_i)$ then s_i is assigned to $M'(s_i)$ in both X and Y. Otherwise, if s_i prefers $M(s_i)$ to $M'(s_i)$ then s_i is assigned to $M(s_i)$ in both X and Y.

(iv) Suppose that $M(s_i)$, $M'(s_i)$ and $M''(s_i)$ are distinct projects. Again, there are six different ways to express s_i's preference over $M(s_i)$, $M'(s_i)$ and $M''(s_i)$. If s_i prefers $M'(s_i)$ to $M''(s_i)$ to $M(s_i)$, then s_i is assigned to $M''(s_i)$ in both X and Y. If s_i prefers $M''(s_i)$ to $M'(s_i)$ to $M(s_i)$, then s_i is assigned to $M'(s_i)$ in both X and Y. We leave it to the reader to verify that in the remaining four cases, s_i is assigned to $M(s_i)$ in both X and Y.

Since s_i is an arbitrary student, it follows that $X = Y$; and thus the second distributive property holds. Since each of M, M' and M'' is an arbitrary stable matching in $\mathcal{M}$, it follows that $(\mathcal{M}, \preceq)$ is a distributive lattice. □

4.4.1 Example

Finally, consider the SPA-S instance I_3 illustrated in Figure 4.4, which admits a total of seven stable matchings, as shown in Table 4.1. The *meet* of matchings M_3 and M_4 is M_2, i.e., $M_2 = M_3 \wedge M_4$. For each student assigned to different projects in M_3 and M_4—namely, s_2, s_4, s_5, s_6, and s_7—the assignment in M_2 corresponds to the better of their projects in M_3 and M_4. Conversely, the *join* of matchings M_3 and M_4 is M_5, i.e., $M_5 = M_3 \vee M_4$. In M_5, each student is assigned to the poorer of their projects in M_3 and M_4.

Students' preferences	**Lecturers' preferences**	**Offers**
s_1: p_1 p_2 p_4 p_3	l_1: s_7 s_9 s_3 s_4 s_5 s_1 s_2 s_6 s_8	p_1, p_2, p_5, p_6
s_2: p_1 p_4 p_3 p_2	l_2: s_6 s_1 s_2 s_5 s_3 s_4 s_7 s_8 s_9	p_3, p_4, p_7, p_8
s_3: p_3 p_1 p_2 p_4		
s_4: p_3 p_2 p_1 p_4		
s_5: p_4 p_3 p_1		
s_6: p_5 p_2 p_7		
s_7: p_7 p_3 p_6		
s_8: p_6 p_8	**Project capacities:** $c_1 = c_3 = 2$; $\forall j \in \{2,4,5,6,7,8\}$, $c_j = 1$	
s_9: p_8 p_2 p_3	**Lecturer capacities:** $d_1 = 4$, $d_2 = 5$	

Figure 4.4: An instance I_3 of SPA-S

Matching	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
M_1	p_1	p_1	p_3	p_3	p_4	p_5	p_7	p_6	p_8
M_2	p_1	p_1	p_3	p_3	p_4	p_5	p_7	p_8	p_2
M_3	p_1	p_1	p_3	p_3	p_4	p_7	p_6	p_8	p_2
M_4	p_1	p_4	p_3	p_1	p_3	p_5	p_7	p_8	p_2
M_5	p_1	p_4	p_3	p_1	p_3	p_7	p_6	p_8	p_2
M_6	p_4	p_3	p_1	p_1	p_3	p_5	p_7	p_8	p_2
M_7	p_4	p_3	p_1	p_1	p_3	p_7	p_6	p_8	p_2

Table 4.1: Instance I_3 admits seven stable matchings.

The Hasse diagram illustrated in Figure 5.6 is a directed graph with each vertex representing a stable matching, and there is a directed edge from vertex M to M' if $M \preceq M'$ and there is no such M^* such that $M \preceq M^* \preceq M'$. We note that all the edges representing precedence implied by transitivity are suppressed in the diagram.

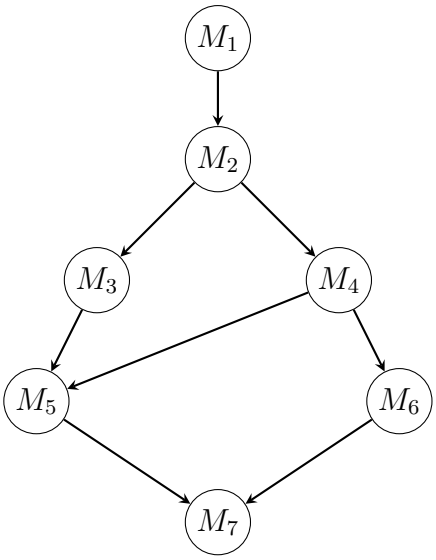

Figure 4.5: Lattice structure for I_3

4.5 Conclusions and future work

In this chapter, we examined the structure of the set of stable matchings in an instance I of SPA-S. We showed that, under a natural dominance relation, this set forms a finite distributive lattice. This result extends known structural properties from the classical Stable Marriage problem to the more complex SPA-S model involving students, projects, and lecturers. In addition, we presented additional properties unique to SPA-S instances which, to the best of our knowledge, have not been previously studied. By establishing the lattice result in SPA-S, we reveal a connection between SPA-S and a broader class of combinatorial problems whose set of solutions form a distributive lattice. This connection enables algorithmic techniques developed in those settings to be applied to related optimisation problems in the SPA-S model.

A natural direction for future work is to explore whether a similar distributive lattice structure exists in the extension of SPA-S that allows ties in preference lists, namely SPA-ST. In this setting, three notions of stability arise: weak stability, strong stability, and super stability. It would be interesting to determine whether a similar characterisation can be developed for the set of strongly stable and super-stable matchings in SPA-ST. The results presented in this chapter provide a starting point for addressing this question and for analysing the complexity of related problems in SPA-S and its extensions. Moreover, the lattice structure described here provides a foundation for further investigation into other polynomial-time characterisations of the set of stable matchings in SPA-S. In the next chapter, we build on the results from this chapter by presenting the *meta-rotation poset*, which provides a compact representation of the set of all stable matchings $\mathcal{M}$ and the lattice for any instance. Moreover, the meta-rotation poset can be constructed in time polynomial in the size of the input.

Chapter 5

Meta-Rotations in SPA-S

5.1 Introduction

In Chapter 4, we proved that the set of all stable matchings $\mathcal{M}$ in a given SPA-S instance forms a distributive lattice, where the student-optimal and lecturer-optimal stable matchings are the two extreme elements of the lattice. In this chapter, we build on that result by introducing *meta-rotations*, which generalises the notion of rotations from the Stable Marriage problem (see Section 2.1.3.2) to the SPA-S setting. We show that each meta-rotation corresponds to a specific set of changes that transforms one stable matching into another within the lattice of stable matchings. Let $\mathcal{M}$ denote the set of all stable matchings in a given SPA-S instance I. We define the *meta-rotation poset* $\Pi(\mathcal{M})$ as the set of meta-rotations in I, together with a partial order defined over them. This poset captures the dependencies between meta-rotations and succinctly encodes the set $\mathcal{M}$. Specifically, we prove that each stable matching in $\mathcal{M}$ corresponds to a unique closed subset of $\Pi(\mathcal{M})$.[1]

5.1.1 Background and motivation

A classical result in lattice theory, Birkhoff's Theorem [16], states that every finite distributive lattice is isomorphic to the collection of closed subsets of some finite partially ordered set (poset). Under this representation, each element of the lattice corresponds to a unique closed subset of the poset, and the meet and join operations in the lattice correspond to the intersection and union of these subsets, respectively. In many combinatorial problems, however, we are typically not given the lattice explicitly. Instead, we are presented with a problem input, for example, a set of agents and their preferences, and the associated distributive lattice arises implicitly from the structure of its set of solutions.

Gusfield and Irving [54] noted that while Birkhoff's Theorem guarantees the existence of a partial order that corresponds to a given lattice, it does not offer a method for constructing this partial

[1] A subset S of a partially ordered set is *closed* (or a *lower set*) if, whenever $x \in S$, all elements y with $y \preceq x$ also belong to S.

order directly from the problem input. Moreover, since the number of stable matchings in an instance can be exponential in the input size, constructing the entire lattice solely to recover the underlying partial order is often infeasible. As a result, Irving and Leather [65] introduced the notions of *rotations* and the *rotation poset*, which provide a compact representation of the set of stable matchings in a given Stable Marriage instance. Crucially, the partial order on rotations can be derived directly from the problem input, without the need to enumerate all stable matchings. They established a one-to-one correspondence between stable matchings and closed subsets of the rotation poset, thereby resolving an open problem posed by Knuth [84]. Further details on the structure of rotations and the construction of the rotation poset are provided in the book by Gusfield and Irving [54], where they also provided an alternative proof using *ring of sets*.

The rotation poset in the Stable Marriage model laid the foundation for efficient algorithms to enumerate all stable marriages, identify all stable pairs, and compute other stable matchings with desirable properties such as the egalitarian or minimum regret stable matching. Since we have already shown in Chapter 4 that the set of stable matchings in any SPA-S instance forms a distributive lattice, it follows from Birkhoff's Theorem that there exists a poset whose closed subsets correspond exactly to these matchings. The goal of this chapter is to show that such a partial order can be explicitly constructed from a given SPA-S instance, by introducing a generalised notion of rotations tailored to SPA-S. Consequently, this structure would enable the design of efficient algorithms for similar problems in SPA-S, such as enumerating all stable matchings and computing other stable matchings that satisfy different optimality criteria.

5.1.2 Contributions and structure of the chapter

As we will demonstrate later, several structural properties that hold in the one-to-one and many-to-many stable matching models do not extend to SPA-S, due to the presence of projects, and the fact that a student may be assigned to different projects offered by the same lecturer in two different stable matchings. These differences motivate the need for a more nuanced definition of rotations that accurately reflects the properties of the SPA-S model.

The main contribution of this chapter is to provide an alternative characterisation of the set of stable matchings in SPA-S through the introduction of meta-rotations, which generalise the classical notion of rotations from SM and HR. We describe how meta-rotations can be identified and eliminated, and show how they can be used to construct the meta-rotation poset, a partial order whose closed subsets are in one-to-one correspondence with the stable matchings of the instance. This poset provides a compact representation of the set of stable matchings, and enables the efficient enumeration and analysis of the set of stable matchings admitted by any SPA-S instance.

In Section 5.2, we provide preliminary definitions and some intuition behind our approach. In Section 5.3, we present results that describe the relationship between student and lecturer prefer-

ences in stable matchings. In Section 5.4, we focus on identifying and eliminating meta-rotations, and describe how they traverse the lattice of stable matchings. Finally, in Section 5.5, we present our main result, which establishes a one-to-one correspondence between the set of stable matchings and the closed subsets of the meta-rotation poset.

5.2 Preliminary definitions

We begin by defining, in Definition 5.2.1, the notion of a valid *next project* for each student who is assigned in some stable matching of a given instance I. This refers to a project in which the student may be assigned to in some other stable matching of I. Then we formally define meta-rotations and describe how to identify an exposed meta-rotation in an arbitrary stable matching (in Definition 5.2.2).

Definition 5.2.1 (Next project). Let M_L denote the lecturer-optimal stable matching in a given SPA-S instance I. For any stable matching $M \neq M_L$, suppose there exists a student s_i such that $M(s_i) \neq M_L(s_i)$. Let $p_j = M(s_i)$ and let l_k be the lecturer offering p_j. Define $w_M(p_j)$ as the worst student assigned to p_j in M, and $w_M(l_k)$ as the worst student assigned to l_k in M.

We define the *next project* for s_i, denoted $s_M(s_i)$, as the first project p on s_i's preference list that appears after p_j and satisfies one of the following conditions, where l is the lecturer offering p:

(i) p is full in M, and l prefers s_i to $w_M(p)$;

(ii) p is undersubscribed in M, l is full in M, and l prefers s_i to $w_M(l)$.

Let $next_M(s_i)$ denote the *next student* for s_i. If p satisfies condition (i), we say $w_M(p)$ is $next_M(s_i)$. If p satisfies condition (ii), then we say that $w_M(l)$ is $next_M(s_i)$. We note that such project p may not always exist. For instance, if M is the lecturer-optimal stable matching, then p does not exist for any student, since each student is assigned in M_L to the worst possible project that they could have in any stable matching.

To illustrate this definition further, consider instance I_1 in Figure 5.1, which admits seven stable matchings, one of which is $M_2 = \{(s_1, p_1), (s_2, p_1), (s_3, p_3), (s_4, p_3), (s_5, p_4), (s_6, p_5), (s_7, p_7), (s_8, p_8), (s_9, p_2)\}$. It can be observed that the first project on s_6's preference list following p_5 (her assigned project in M_2) is p_2, which is full in M_2. However, l_1 (the lecturer offering p_2) prefers the worst student in $M_2(p_2)$, namely s_9, to s_6. Proceeding to the next project, p_7, which is full in M_2, it is clear that l_2 prefers s_6 to the worst student in $M_2(p_7)$, namely s_7. Therefore, $s_M(s_6) = p_7$

and $next_{M_2}(s_6) = s_7$. Similarly, p_6 is the first project on s_7's preference list that is undersubscribed in M_2, and l_1 prefers s_7 to the worst student in $M_2(l_1)$, namely s_6. Thus, $s_M(s_7) = p_6$ and $next_{M_2}(s_7) = s_6$.

Students' preferences	**Lecturers' preferences**	**Offers**
s_1: p_1 p_2 p_4 p_3	l_1: s_7 s_9 s_3 s_4 s_5 s_1 s_2 s_6 s_8	p_1, p_2, p_5, p_6
s_2: p_1 p_4 p_3 p_2	l_2: s_6 s_1 s_2 s_5 s_3 s_4 s_7 s_8 s_9	p_3, p_4, p_7, p_8
s_3: p_3 p_1 p_2 p_4		
s_4: p_3 p_2 p_1 p_4		
s_5: p_4 p_3 p_1		
s_6: p_5 p_2 p_7		
s_7: p_7 p_3 p_6		
s_8: p_6 p_8	**Project capacities:** $c_1 = c_3 = 2$; $\forall j \in \{2,4,5,6,7,8\}$, $c_j = 1$	
s_9: p_8 p_2 p_3	**Lecturer capacities:** $d_1 = 4$, $d_2 = 5$	

Figure 5.1: An instance I_1 of SPA-S

Matching	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
M_1	p_1	p_1	p_3	p_3	p_4	p_5	p_7	p_6	p_8
M_2	p_1	p_1	p_3	p_3	p_4	p_5	p_7	p_8	p_2
M_3	p_1	p_1	p_3	p_3	p_4	p_7	p_6	p_8	p_2
M_4	p_1	p_4	p_3	p_1	p_3	p_5	p_7	p_8	p_2
M_5	p_1	p_4	p_3	p_1	p_3	p_7	p_6	p_8	p_2
M_6	p_4	p_3	p_1	p_1	p_3	p_5	p_7	p_8	p_2
M_7	p_4	p_3	p_1	p_1	p_3	p_7	p_6	p_8	p_2

Table 5.1: Instance I_1 admits seven stable matchings.

Definition 5.2.2 (Exposed Meta-Rotation). Let M be a stable matching, and let $\rho = \{(s_0, p_0), (s_1, p_1), \ldots, (s_{r-1}, p_{r-1})\}$ be an ordered list of student–project pairs in M, where $r \geq 2$. For each $t (0 \leq t \leq r-1)$, suppose that s_t is the worst student assigned to project p_t in M, and $s_{t+1} = next_M(s_t)$ (with indices taken modulo r). Then ρ is called an exposed meta-rotation in M. Moreover, if a pair $(s, p) \in \rho$, we say that $s \in \rho$ (or equivalently, $p \in \rho$).

Note that in any exposed meta-rotation ρ of a stable matching M, each student and project that appears in ρ is part of an assigned pair in M, and each appears exactly once in ρ. This is because, in M, each project has a unique worst student among those assigned to it, and the definition of ρ includes precisely one such student–project pair. Furthermore, the set of all meta-rotations in I

consists precisely of those ordered sets of pairs that are exposed in at least one stable matching $M \in \mathcal{M}$.

Definition 5.2.3 (Meta-rotation Elimination). Given a stable matching M and an exposed meta-rotation ρ in M, we denote by M/ρ the matching obtained by assigning each student $s \in \rho$ to project $s_M(s)$, while keeping the assignments of all other students unchanged. This transition from M to M/ρ is referred to as the *elimination* of ρ from M.

5.2.1 Justification for the meta-rotation definition

In both SM and HR, an exposed rotation ρ in a stable matching M is defined as a sequence of pairs such that performing a cyclic shift yields a new stable matching M/ρ: in SM, each woman is assigned to the next man in the sequence, and in HR, each hospital is assigned to the next resident. Specifically, in HR, if some resident r, who is assigned in a stable matching M, has a *next* hospital h on their preference list and is part of an exposed rotation ρ, then r swaps places with the least preferred resident currently assigned to h in M, forming the new matching M/ρ. Moreover, by the Rural Hospitals Theorem for HR (see Theorem 2.2.1), if some hospital h is undersubscribed in one stable matching, then it is assigned the same set of residents across all stable matchings.

However, as we noted in Chapter 4, these properties do not extend to SPA-S for projects or lecturers that are undersubscribed. In SPA-S, the number of students assigned to a project may vary across stable matchings. Consequently, a project that is part of an exposed meta-rotation ρ in a given stable matching M may not necessarily appear in the resulting stable matching M/ρ. For example, in instance I_3 from Figure 5.1, the pairs $\{(s_6, p_5), (s_7, p_7)\}$ form an exposed meta-rotation in M_2. Here, project p_5 is full in M_2 but becomes undersubscribed in M_3. Clearly, neither p_5 nor its lecturer l_1 (who offers p_5) have the same set of assigned students in M_2 and M_3. Nevertheless, by the *Unpopular Projects Theorem* (see Theorem 4.2.1), the total number of students assigned to each lecturer remains the same across all stable matchings.

To address these differences, our definition of meta-rotations explicitly accounts for whether each project is full or undersubscribed in the stable matching of interest. Suppose a student s_i, assigned to some project in a stable matching M, has p_j as their next possible project. Whether s_i can be assigned to p_j in another stable matching depends on the status of p_j in M as well as the preference of the lecturer l_k who offers it. If p_j is full in M, then the assignment of s_i to p_j is possible only if l_k prefers s_i to the worst student in $M(p_j)$; in this case, s_i takes the place of that student. If p_j is undersubscribed in M, then the assignment is possible only if l_k prefers s_i to the worst student in $M(l_k)$; here, s_i is assigned to p_j and the least preferred student in $M(l_k)$ is removed. These conditions ensure that each such assignment yields a new matching that is stable.

5.3 Structural results involving stable matchings

In this section, we present new results on stable matchings in a SPA-S instance, providing insight into how the assignment of a student to different projects in two stable matchings affects the preferences of the involved lecturers. Throughout, let l_k denote the lecturer offering project p_j.

In Lemma 5.3.1, we show that for any two stable matchings M and M' where M dominates M', if $(s_i, p_j) \in M' \setminus M$ and p_j is full in M, then the worst student in $M(p_j)$ is not assigned to p_j in M'. If instead p_j is undersubscribed in M, then the worst student in $M(l_k)$, does not appear in $M'(l_k)$. In Lemma 5.3.2, we show that if s_i is assigned to different projects in M and M', and is assigned to p_j in M', then l_k prefers s_i to the worst student in $M(p_j)$ when p_j is full in M, and l_k prefers s_i to the worst student in $M(l_k)$ when p_j is undersubscribed. Finally, in Lemma 5.3.3, we show that if M dominates M', and some student s_i is assigned to p_j in M' but to a different project in M, then if p_j is undersubscribed in M, then l_k must be full in M.

Lemma 5.3.1. *Let M and M' be two stable matchings where M dominates M'. Suppose there exists a student s_i who is assigned to different projects in M and M', with s_i assigned to project p_j in M'. Then the following hold:*

(i) If p_j is full in M, the worst student in $M(p_j)$ is not in $M'(p_j)$.

(ii) If p_j is undersubscribed in M, the worst student in $M(l_k)$ is not in $M'(l_k)$.

Proof. Let s_i be some student assigned to different projects in M and M', such that $s_i \in M'(p_j) \setminus M(p_j)$, and l_k offers p_j. Let s_z be the worst student in $M(p_j)$, and suppose for a contradiction that $s_z \in M(p_j) \cap M'(p_j)$. Consider case (i) where p_j is full in M. Since $s_i \in M'(p_j) \setminus M(p_j)$ and $|M(p_j)| \geq |M'(p_j)|$, there exists some student $s_t \in M(p_j) \setminus M'(p_j)$. Moreover, since s_z is the worst student in $M(p_j)$, l_k prefers s_t to s_z. Since M dominates M', s_t prefers M to M'. Regardless of whether p_j is full or undersubscribed in M', the pair (s_t, p_j) blocks M', leading to a contradiction. Therefore, case (i) holds.

Now consider case (ii) where p_j is undersubscribed in M. Let s_z be the worst student in $M(l_k)$, and suppose for a contradiction that $s_z \in M(l_k) \cap M'(l_k)$. First, suppose that $|M(p_j)| \geq |M'(p_j)|$. Since p_j is undersubscribed in M, it follows that p_j is undersubscribed in M'. Given that $s_i \in M'(p_j) \setminus M(p_j)$, there exists some student $s_r \in M(p_j) \setminus M'(p_j)$. Furthermore, s_r prefers M to M', and either $s_r = s_z$ or l_k prefers s_r to s_z. If $s_r = s_z$, then $s_r \in M'(l_k)$ and, since p_j is undersubscribed in M', the pair (s_r, p_j) blocks M', leading to a contradiction. If instead $s_r \neq s_z$, then l_k prefers s_r to s_z, since s_z is the worst student in $M(l_k)$. However, given that s_r prefers M to M', p_j is undersubscribed in M', and l_k prefers s_r to s_z, the pair (s_r, p_j) blocks M', again leading to a contradiction.

Now, suppose that $|M'(p_j)| > |M(p_j)|$. Since $|M(l_k)| = |M'(l_k)|$, there exists some project $p_t \in P_k$ such that $|M(p_t)| > |M'(p_t)|$, meaning p_t is undersubscribed in M'. Consequently, there exists a

student $s_t \in M(p_t) \setminus M'(p_t)$ who prefers M to M'. If $s_t = s_z$, then $s_t \in M'(l_k)$ and, since p_t is undersubscribed in M', the pair (s_t, p_t) blocks M', leading to a contradiction. Otherwise, since s_z is the worst student in $M(l_k)$, it follows that l_k prefers s_t to s_z. Given that s_t prefers M to M', p_t is undersubscribed in M', and l_k prefers s_t to s_z, the pair (s_t, p_t) blocks M', leading to a contradiction. Hence, our claim holds. □

Lemma 5.3.2. *Let M and M' be two stable matchings in I such that M dominates M'. Suppose that a student s_i is assigned to different projects in M and M', such that s_i is assigned to project p_j in M', where l_k offers p_j. Then the following conditions hold:*

(i) If p_j is full in M, then l_k prefers s_i to the worst student in $M(p_j)$.

(ii) If p_j is undersubscribed in M, then l_k prefers s_i to the worst student in $M(l_k)$.

Proof. Let M and M' be two stable matchings in I, where M dominates M'. Suppose that some student s_i is assigned to project p_j in M', where l_k offers p_j (and possibly l_k offers $M(s_i)$). Consider case (i), where p_j is full in M. Let s_z be the worst student in $M(p_j)$, and suppose for a contradiction that l_k prefers s_z to s_i. By Lemma 5.3.1, it follows that $s_z \notin M'(p_j)$, meaning $s_z \in M(p_j) \setminus M'(p_j)$. Since M dominates M', s_z prefers p_j to $M'(s_z)$. If p_j is full in M', then the pair (s_z, p_j) blocks M', since l_k prefers s_z to some student in $M'(p_j)$, namely s_i. Similarly, if p_j is undersubscribed in M', (s_z, p_j) also blocks M', since l_k prefers s_z to some student in $M'(l_k)$, namely s_i. This leads to a contradiction. Hence, l_k prefers s_i to s_z, and case (i) holds.

Consider case (ii), where p_j is undersubscribed in M. Now, suppose for a contradiction that l_k prefers the worst student in $M(l_k)$ to s_i. This means that l_k prefers every student in $M(l_k)$ to s_i. First, suppose that $|M(p_j)| \geq |M'(p_j)|$. Then, p_j is also undersubscribed in M'. Since $M(p_j)$ contains at least as many students as $M'(p_j)$, there must be some student $s_r \in M(p_j) \setminus M'(p_j)$ (Readers may recall that $s_i \in M'(p_j) \setminus M(p_j)$). Additionally, s_r prefers M to M', since M dominates M'. Given that $s_r \in M(l_k)$ and s_r is either the worst student in $M(l_k)$ or better, it follows that l_k prefers s_r to s_i. However, since p_j is undersubscribed in M' and l_k prefers s_r to some student in $M'(l_k)$ (namely s_i), the pair (s_r, p_j) blocks M', leading to a contradiction.

Now, suppose instead that $|M(p_j)| < |M'(p_j)|$. Since $|M(l_k)| = |M'(l_k)|$, there exists some other project $p_t \in P_k$ such that $|M'(p_t)| < |M(p_t)|$. This means p_t is undersubscribed in M' and there exists some student $s_t \in M(p_t) \setminus M'(p_t)$, that is, $s_t \in M(l_k)$. Moreover, s_t prefers M to M'. Since p_t is undersubscribed in M' and l_k prefers s_t to some student in $M'(l_k)$ (namely s_i), the pair (s_t, p_t) blocks M', contradicting the stability of M'. Thus, we reach a contradiction in both scenarios, completing the proof for case (ii). Therefore, the lemma holds. □

Lemma 5.3.3. *Let M and M' be two stable matchings where M dominates M'. Suppose that a student s_i is assigned to different projects in M and M', with s_i assigned to project p_j in M'. If p_j is undersubscribed in M then l_k is full in M.*

Proof. Let M and M' be two stable matchings where M dominates M'. Suppose s_i is some student assigned to different projects in M and M', such that s_i is assigned to p_j in M', and l_k offers p_j (possibly l_k also offers $M(s_i)$). Now, suppose for a contradiction that both p_j and l_k are undersubscribed in M. Since p_j is offered by an undersubscribed lecturer l_k, it follows from the Unpopular Projects Theorem (see Theorem 4.2.1) that the same number of students are assigned to p_j in M and M'. Therefore, since $s_i \in M'(p_j) \setminus M(p_j)$, there exists some student s_z such that $s_z \in M(p_j) \setminus M'(p_j)$. Moreover, both p_j and l_k are undersubscribed in M', since $|M(p_j)| = |M'(p_j)|$ and $|M(l_k)| = |M'(l_k)|$. Since M dominates M', s_z prefers p_j to $M'(s_z)$. However, since p_j and l_k are both undersubscribed in M', the pair (s_z, p_j) blocks M', a contradiction. Hence, our claim holds. □

5.4 Exposing and eliminating all meta-rotations

In this section, we present key lemmas to show that by successively identifying and eliminating exposed meta-rotations, we obtain another stable matching of a given instance. First, we recall the following results from Chapter 4, which will be used in the proofs for this section:

Lemma 5.4.1. *Let M and M' be two stable matchings in a given instance I. Suppose a student s_i is assigned to different projects in M and M', and that in M', s_i is assigned to a project p_j offered by lecturer l_k. Suppose further that s_i prefers M to M'. Then:*

(a) If there exists a student in $M(p_j) \setminus M'(p_j)$, then l_k prefers s_i to each student in $M(p_j) \setminus M'(p_j)$.

(b) If p_j is undersubscribed in M, then l_k prefers s_i to each student in $M(l_k) \setminus M'(l_k)$.

Lemma 5.4.2. *Let M and M' be two stable matchings in I. If a student s_i is assigned in M and M' to different projects offered by the same lecturer l_k, and s_i prefers M to M', then there exists some student $s_r \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s_r to s_i. Thus, $M(l_k) \neq M'(l_k)$.*

5.4.1 Meta-rotations

Let $\rho = \{(s_0, p_0), (s_1, p_1), \ldots, (s_{r-1}, p_{r-1})\}$ be an exposed meta-rotation in a stable matching M of some SPA-S instance I, and consider any pair $(s_t, p_t) \in \rho$. Let the next project for s_t be $s_M(s_t)$, and suppose that there exists some project p_z that lies strictly between p_t and $s_M(s_t)$ in s_t's preference list. Then, by Lemma 5.4.3, the pair (s_t, p_z) does not appear in any stable matching of I, and hence is not a stable pair.

In Lemma 5.4.4, we show that in any SPA-S instance, every stable matching other than the lecturer-optimal matching M_L contains at least one exposed meta-rotation. In Lemma 5.4.5, we show that if, in the construction of M/ρ, a student becomes assigned to a lecturer l_k, then l_k simultaneously loses a student from $M(l_k)$. Finally, in Lemma 5.4.6, we prove that if a meta-rotation ρ is exposed in a stable matching M, then the matching M/ρ, obtained by eliminating ρ from M, is also stable, and that M dominates M/ρ.

Lemma 5.4.3. *Let $\rho = \{(s_0, p_0), (s_1, p_1), \ldots, (s_{r-1}, p_{r-1})\}$ be an exposed meta-rotation in a stable matching M for instance I. Suppose that for some student s_t (where $0 \leq t \leq r-1$), there exists a project p_z such that s_t prefers p_t to p_z, and prefers p_z to $s_M(s_t)$. Then the pair (s_t, p_z) is not a stable pair — that is, it is not part of any stable matching of I.*

Proof. Let M be a stable matching in which the meta-rotation ρ is exposed, and suppose that $(s_i, p_j) \in \rho$. Suppose there exists a project p_z on s_i's preference list such that s_i prefers p_j to p_z, and prefers p_z to $s_M(s_i)$. Let l_z be the lecturer who offers p_z, and possibly also offers $s_M(s_i)$. Suppose for contradiction that there exists another stable matching M' in which s_i is assigned to p_z, that is, $s_i \in M'(p_z) \setminus M(p_z)$. Then s_i prefers M to M'. Since $p_z \neq s_M(s_i)$, by definition of $s_M(s_i)$, one of the following conditions must hold in M:

(i) both p_z and l_z are undersubscribed,

(ii) p_z is full and l_z prefers the worst student in $M(p_z)$ to s_i, or

(iii) p_z is undersubscribed, l_z is full, and l_z prefers the worst student in $M(l_z)$ to s_i.

Case (i): Suppose both p_z and l_z are undersubscribed in M. Then l_z is undersubscribed in M' since $|M(l_z)| = |M'(l_z)|$. Moreover, by the Unpopular Projects Theorem (see Theorem 4.2.1), since p_z is offered by an undersubscribed lecturer l_z, then $|M(p_z)| = |M'(p_z)|$, meaning p_z is undersubscribed in M'. Since $s_i \in M'(p_z) \setminus M(p_z)$, there must exist a student $s_z \in M(p_z) \setminus M'(p_z)$. If s_z prefers M to M', then (s_z, p_z) blocks M', as p_z and l_z are undersubscribed in M'. Therefore, s_z prefers M' to M. Now, by the first part of Lemma 5.4.1, since s_z prefers M' to M and $s_i \in M'(p_z) \setminus M(p_z)$, then l_z prefers s_z to s_i. However, by the same lemma, since s_i prefers M to M' and $s_z \in M(p_z) \setminus M'(p_z)$, then l_z prefers s_i to s_z. This gives a direct contradiction, as l_z cannot simultaneously prefer s_i to s_z and s_z to s_i. Hence, case (i) cannot occur.

Case (ii): Suppose p_z is full in M and l_z prefers the worst student in $M(p_z)$ to s_i. Since $s_i \in M'(p_z) \setminus M(p_z)$ and p_z is full in M, there exists some student $s_z \in M(p_z) \setminus M'(p_z)$. Thus, l_z prefers s_z to s_i. However, by Lemma 5.4.1, since s_i prefers M to M' and $s_z \in M(p_z) \setminus M'(p_z)$, it follows that l_z prefers s_i to s_z. This gives a direct contradiction, as l_z cannot simultaneously prefer s_i to s_z and s_z to s_i. Therefore, case (ii) cannot occur.

Case (iii): Suppose p_z is undersubscribed in M, l_z is full in M, and l_z prefers the worst student in $M(l_z)$ to s_i. This implies that l_z prefers each student in $M(l_z)$ to s_i. We claim that there exists

some student $s_z \in M(l_z) \setminus M'(l_z)$. If $s_i \in S_z(M, M')$, meaning s_i is assigned to different projects offered by l_z in both M and M', then by Lemma 5.4.2, there exists some $s \in M'(l_z) \setminus M(l_z)$. Consequently, there must exist some $s_z \in M(l_z) \setminus M'(l_z)$, since $|M(l_k)| = |M'(l_k)|$. The same conclusion holds if $s_i \in M'(l_z) \setminus M(l_z)$. Thus, it follows that l_z prefers s_z to s_i. However, by Lemma 5.4.1, since s_i prefers M to M' and p_z is undersubscribed in M, we have that l_z prefers s_i to s_z. This gives a direct contradiction, as l_z cannot simultaneously prefer s_z to s_i and s_i to s_z. Therefore, case (iii) cannot occur.

Since all possible cases lead to a contradiction, the pair (s_i, p_z) cannot belong to any stable matching of I, completing the proof. □

The following corollary follows immediately from Lemma 5.4.3:

Corollary 5.4.1. *Let M be a stable matching in I, and let s_i be a student for whom $s_M(s_i)$ exists. Suppose that s_i prefers $M(s_i)$ to some project p_z offered by lecturer l_z, and prefers p_z to $s_M(s_i)$. If both p_z and l_z are undersubscribed in M, then the pair (s_i, p_z) does not appear in any stable matching of I.*

Lemma 5.4.4. *Let M be a stable matching in an instance of* SPA-S*, and suppose $M \neq M_L$, where M_L is the lecturer-optimal stable matching. Then there exists at least one meta-rotation that is exposed in M.*

Proof. Let M be a stable matching in an instance I of SPA-S, and let M_L be the lecturer-optimal stable matching. Clearly, M dominates M_L. Since $M \neq M_L$, there exists some student s_{i_0}, who is assigned to different projects in M and M_L. Suppose that s_{i_0} is assigned to p_{j_0} in M and assigned to p_{t_0} in M_L, where l_t offers p_{t_0} (possibly l_t offers both p_{j_0} and p_{t_0}). Clearly, s_{i_0} prefers p_{j_0} to p_{t_0}. Furthermore, p_{t_0} is either (i) undersubscribed in M or (ii) full in M. In both cases, we will prove that $s_M(s_{i_0})$ exists, which in turn proves the existence of $next_M(s_{i_0})$.

First, suppose that p_{t_0} is undersubscribed in M. By Lemma 5.3.2, l_t prefers s_{i_0} to the worst student in $M(l_t)$. Furthermore, by Lemma 5.3.3, if p_{t_0} is undersubscribed in M, then l_t must be full in M. Given that s_{i_0} prefers p_{j_0} to p_{t_0}, p_{t_0} is undersubscribed in M, l_t is full in M, and l_t prefers s_{i_0} to the worst student in $M(l_t)$, it follows that $s_M(s_{i_0})$ exists. Now, consider case (ii), where p_{t_0} is full in M. Since s_{i_0} is assigned to p_{t_0} in M_L and p_{t_0} is full in M, by Lemma 5.3.2, we have that l_t prefers s_{i_0} to the worst student in $M(p_{t_0})$. Since these condition hold, $s_M(s_{i_0})$ exists, and consequently, $next_M(s_{i_0})$ exists.

Let $next_M(s_{i_0}) = s_{i_1}$. By definition, s_{i_1} is either the worst student assigned to p_{t_0} in M (if p_{t_0} is full in M), or the worst student assigned to l_t in M (if p_{t_0} is undersubscribed in M). In either case, l_t prefers s_{i_0} to s_{i_1}. Furthermore, since s_{i_0} is assigned to p_{j_0} in M and to p_{t_0} in M_L, it follows from Lemma 5.3.1 that the worst student in $M(p_{t_0})$ is not in $M_L(p_{t_0})$ (if p_{t_0} is full in M), and the

worst student in $M(l_t)$ is not in $M_L(l_t)$ (if p_{t_0} is undersubscribed in M). Therefore, s_{i_1} is assigned to different projects in M and M_L. Let $p_{j_1} = M(s_{i_1})$, where l_t offers p_{j_1} (possibly $p_{t_0} = p_{j_1}$). Let $p_{t_1} = M_L(s_{i_1})$, and let l_{t_1} be the lecturer who offers p_{t_1} (possibly $l_t = l_{t_1}$). Clearly, s_{i_1} prefers p_{j_1} to p_{t_1}. Again, it follows that p_{t_1} is either (i) undersubscribed in M or (ii) full in M. Following a similar argument as before, we will prove that both $s_M(s_{i_1})$ and $next_M(s_{i_1})$ exist.

First, suppose that p_{t_1} is undersubscribed in M. By Lemma 5.3.2, l_{t_1} prefers s_{i_1} to the worst student in $M(l_{t_1})$. Furthermore, by Lemma 5.3.3, if p_{t_1} is undersubscribed in M, then l_{t_1} must be full in M. Given that s_{i_1} prefers p_{j_1} to p_{t_1}, p_{t_1} is undersubscribed in M, l_{t_1} is full in M, and l_{t_1} prefers s_{i_1} to the worst student in $M(l_{t_1})$, it follows that $s_M(s_{i_1})$ exists. Now, consider case (ii), where p_{t_1} is full in M. Since s_{i_1} is assigned to p_{t_1} in M_L and p_{t_1} is full in M, by Lemma 5.3.2, we have that l_{t_1} prefers s_{i_1} to the worst student in $M(p_{t_1})$. Since this condition holds, $s_M(s_{i_1})$ exists, and consequently, $next_M(s_{i_1})$ exists.

Let $next_M(s_{i_1}) = s_{i_2}$. By definition, s_{i_2} is either the worst student assigned in $M(p_{t_1})$ if p_{t_1} is full in M, or the worst student in $M(l_{t_1})$ if p_{t_1} is undersubscribed in M. In either case, l_{t_1} prefers s_{i_1} to s_{i_2}. Furthermore, since s_{i_1} is assigned to p_{j_1} in M and to p_{t_1} in M_L, it follows from Lemma 5.3.1 that the worst student in $M(p_{t_1})$ is not in $M_L(p_{t_1})$ (if p_{t_1} is full in M), and the worst student in $M(l_{t_1})$ is not in $M_L(l_{t_1})$ (if p_{t_1} is undersubscribed in M). Therefore, s_{i_2} is assigned to different projects in M and M_L. Let $p_{j_2} = M(s_{i_2})$, where l_{t_1} offers p_{j_2} (possibly $p_{j_2} = p_{t_1}$). Let $p_{t_2} = M_L(s_{i_2})$, and let l_{t_2} be the lecturer who offers p_{t_2}. Clearly, s_{i_2} prefers p_{j_2} to p_{t_2}. Again, it follows that p_{t_2} is either (i) undersubscribed in M or (ii) full in M. Following a similar argument as in the previous paragraphs, both $s_M(s_{i_2})$ and $next_M(s_{i_2})$ exist.

By continuing this process, we observe that each identified student-project pair (s_i, p_j) in M leads to another pair in M, which in turn leads to another pair, and so forth, thereby forming a sequence of pairs $(s_{i_0}, p_{j_0}), (s_{i_1}, p_{j_1}), \ldots$ within M such that s_{i_1} is $next_M(s_{i_0})$, s_{i_2} is $next_M(s_{i_1})$, and so on. Moreover, each student that we identify is assigned to different projects in M and M_L, and prefers their assigned project in M to M_L. Given that the number of students in M is finite, this sequence cannot extend indefinitely and must eventually terminate with a pair in M that we have previously identified.

Suppose that $(s_{i_{r-1}}, p_{j_{r-1}})$ is the final student-project pair identified in this sequence, let s_{i_r} be next$_M(s_{i_{r-1}})$, and let $M(s_{i_r})$ be p_{j_r}. It follows that s_{i_r} must have appeared earlier in the sequence. Otherwise, we would need to extend the sequence by including the pair, (s_{i_r}, p_{j_r}), contradicting the assumption that $(s_{i_{r-1}}, p_{j_{r-1}})$ is the last pair identified in the sequence. Therefore, at some point, a student-project pair must reappear in the sequence, and when this occurs, the process terminates. As an example, suppose that the sequence starts with (s_{i_0}, p_{j_0}), and that the last pair (s_{i_r}, p_{j_r}) satisfies $s_{i_r} = s_{i_1}$. Then, the subsequence $\{(s_{i_1}, p_{j_1}), (s_{i_2}, p_{j_2}), \ldots, (s_{i_{r-1}}, p_{j_{r-1}})\}$ forms an exposed meta-rotation in M, as illustrated in Figure 5.2.

□

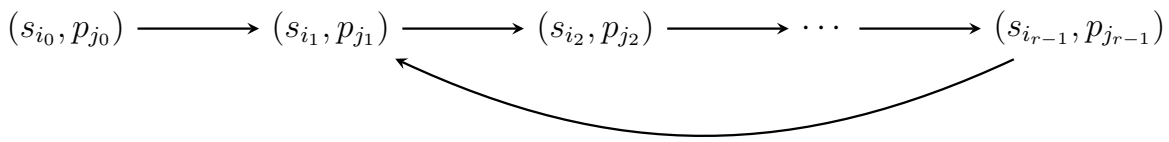

Figure 5.2: Exposed meta-rotation in M.

5.4.2 Identifying an exposed meta-rotation

The proof of Lemma 5.4.4 describes a method for identifying an exposed meta-rotation in any given stable matching M for some SPA-S instance I. Given a stable matching M, define a directed graph $H(M)$ with a vertex for each student s_i who is assigned different projects in M and M_L. For each such student s_i, add a directed edge from s_i to $next_M(s_i)$, which, from the previous proof, must also be a vertex in $H(M)$. Clearly, every vertex in $H(M)$ has exactly one outgoing edge because each student s_i in $H(M)$ has exactly one $next_M(s_i)$. Since the number of vertices (students) is finite, $H(M)$ must contain at least one directed simple cycle. This cycle corresponds to the set of students involved in an exposed meta-rotation in M; for any student s_i in the cycle, $(s_i, M(s_i))$ is a pair in the exposed meta-rotation.

To identify an exposed meta-rotation in M, start from any student s_i and traverse the directed path in $H(M)$ until some student is visited twice. Let s_k be the first student that appears twice in the traversal. Then, the students involved in the exposed meta-rotation are those encountered from the first occurrence of s_k up to and including the student immediately before its second occurrence in the sequence.

Corollary 5.4.2. *Let M be a stable matching that differs from the lecturer-optimal stable matching M_L. Consider the directed graph $H(M)$, whose vertex set consists precisely of those students s_i assigned to different projects in M and M_L. The edge set of $H(M)$ consists of all directed edges $(s_i, next_M(s_i))$, one for each vertex s_i in the graph. It follows that*

- *Each vertex $s_i \in H(M)$ has exactly one outgoing edge.*
- *Beginning from any vertex $s_i \in H(M)$, there exists a unique directed path in $H(M)$ that terminates at a vertex corresponding to the last student appearing in an exposed meta-rotation ρ in M.*
- *Every student in $H(M)$ either belongs to exactly one exposed meta-rotation in M or lies on the path leading to exactly one meta-rotation.*

Example: Consider instance I_2 in Figure 5.3, where the student-optimal stable matching is $M = \{(s_1, p_1), (s_2, p_3), (s_3, p_2), (s_4, p_4)\}$, and the lecturer-optimal stable matching is $M_L = \{(s_1, p_2), (s_2, p_4),$

$(s_3, p_1), (s_4, p_3)\}$. Each student is assigned to different projects in M and M_L, and for each student, we have: $next_M(s_1) = s_3,\ next_M(s_2) = s_4,\ next_M(s_3) = s_1,\ next_M(s_4) = s_1$. The directed graph $H(M)$ corresponding to M is shown in Figure 5.4. Starting at s_2, the sequence of visited students is: $s_2 \to s_4 \to s_1 \to s_3 \to s_1$. Since s_1 appears twice, the first cycle in this sequence is determined by the students from the first occurrence of s_1 up to (but not including) its second occurrence. Thus, the students forming the meta-rotation are s_1 and s_3, and the corresponding meta-rotation exposed in M is $\rho = \{(s_1, p_1), (s_3, p_2)\}$.

Students' preferences	Lecturers' preferences	offers
s_1: p_1 p_2	l_1: s_1 s_3	p_2
s_2: p_3 p_4	l_2: s_2 s_4	p_4
s_3: p_2 p_1	l_3: s_3 s_4 s_1	p_1
s_4: p_4 p_1 p_3	l_4: s_4 s_2 s_1	p_3
	Project capacities: $\forall c_j = 1$	
	Lecturer capacities: $\forall d_k = 1$	

Figure 5.3: An instance I_2 of SPA-S

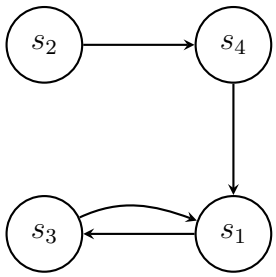

Figure 5.4: Graph $H(M)$ for M

We observe that a student s_i may be assigned different projects in M and M_L without being part of an exposed meta-rotation ρ in M. In such a case, if there exists a directed path from s_i to some student involved in ρ, we say that s_i *leads to* ρ. For instance, $s_4 \in M_L(l_4) \setminus M(l_4)$ and $s_4 \notin \rho$, so s_4 *leads to* ρ.

Lemma 5.4.5. *Let M be a stable matching in I different from the lecturer-optimal matching M_L and let ρ be an exposed meta-rotation in M. If some student $s_i \in \rho$ such that $s_M(s_i)$ is offered by lecturer l_k, then there exists some other student $s_z \in M(l_k)$ such that l_k prefers s_i to s_z, $s_z \in \rho$, and $s_M(s_z)$ is offered by a lecturer different from l_k.*

Proof. Let M be a stable matching with an exposed meta-rotation ρ. Suppose there exists some student $s_{i_0} \in \rho$, such that $s_M(s_{i_0})$ is offered by lecturer l_k. Without loss of generality, suppose that (s_{i_0}, p_{j_0}) is the first pair in ρ. Now suppose for a contradiction that there exists no student $s_z \in M(l_k)$, such that $s_z \in \rho$ and $s_M(s_z)$ is offered by a lecturer different from l_k. The reader

may recall that for every student $s_i \in \rho$, there is a corresponding $s_M(s_i)$ and a $next_M(s_i)$, with $next_M(s_i)$ being a student in ρ.

Since $s_{i_0} \in \rho$, there exists a student $s_{i_1} \in \rho$ where $s_{i_1} = \text{next}_M(s_{i_0})$ and, by definition of $next_M(s_{i_0})$, l_k prefers s_{i_0} to s_{i_1}. Hence, $s_M(s_{i_1})$ exists and by our assumption, $s_M(s_{i_1})$ is offered by l_k. Similarly, since $s_{i_1} \in \rho$, there exists a student $s_{i_2} \in \rho$ with $s_{i_2} = \text{next}_M(s_{i_1})$ and l_k prefers s_{i_1} to s_{i_2}. Again, by our assumption, $s_M(s_{i_2})$ is also offered by l_k. Continuing in this manner, we obtain a sequence of student-project pairs $(s_{i_0}, p_{j_0}), (s_{i_1}, p_{j_1}), (s_{i_2}, p_{j_2}), \ldots, (s_{i_{r-1}}, p_{j_{r-1}}), (s_{i_r}, p_{j_r})$ in ρ such that for each t with $0 \leq t < r$:

- $s_{i_{t+1}} = \text{next}_M(s_{i_t})$,
- l_k prefers s_{i_t} to $s_{i_{t+1}}$, and
- $s_M(s_{i_{t+1}})$ is offered by l_k.

Since ρ is finite, this sequence cannot continue indefinitely and we would identify some student-project pair that appeared earlier in the sequence. Without loss of generality, let (s_{i_r}, p_{j_r}) be the first pair to reappear in the sequence. By construction, s_{i_r} is $next_M(s_{i_{r-1}})$, l_k prefers $s_{i_{r-1}}$ to s_{i_r}, and $s_M(s_{i_r})$ is offered by l_k. Clearly, $s_{i_r} \neq s_{i_{r-1}}$. Therefore, s_{i_r} must have appeared earlier in the sequence before $s_{i_{r-1}}$. However, since s_{i_r} appears earlier in the sequence, then s_{i_r} must be some student that l_k prefers to $s_{i_{r-1}}$, that is, l_k prefers s_{i_r} to $s_{i_{r-1}}$. This yields a contradiction since we assume that l_k prefers $s_{i_{r-1}}$ to s_{i_r}. Therefore, our claim holds, and there must exist at least one student $s_z \in M(l_k)$, such that $s_z \in \rho$ and $s_M(s_z)$ is offered by a lecturer different from l_k. □

Lemma 5.4.6. *If ρ is a meta-rotation exposed in a stable matching M, then the matching obtained by eliminating ρ from M, denoted as M/ρ, is a stable matching. Furthermore, M dominates M/ρ.*

Proof. Let M be a stable matching in which ρ is exposed, and let M' be the matching obtained by eliminating ρ from M, that is, $M' = M/\rho$. First, note that any student assigned to different projects in M and M' must be in ρ, since by definition, each student not in ρ remains assigned to the same project in M and M'. Also, by eliminating ρ from M, each student $s_i \in \rho$ is no longer assigned to $M(s_i)$ but is assigned to $s_M(s_i)$ in M'. Consequently, each student in M' is assigned exactly one project, and no student is multiply assigned.

Next, consider any project p_j where $M'(p_j) \neq M(p_j)$. If p_j is full in M, then the elimination of ρ from M results in p_j losing exactly one student—the worst student in $M(p_j)$—and gaining exactly one student in $M'(p_j)$. Hence, p_j remains full in M' and $|M(p_j)| = |M'(p_j)|$. If p_j is undersubscribed in M, then the lecturer l_k who offers p_j loses the worst student in $M(l_k)$, while p_j gains exactly one student in M'. Consequently, p_j remains either undersubscribed in M' or becomes full in M', that is, $|M(p_j)| \leq |M'(p_j)|$. Therefore, no project is oversubscribed in M'.

Now we show that no lecturer is oversubscribed in M'. Since ρ is exposed in M, there exists some student $s_i \in \rho$. Let l be the lecturer who offers $s_M(s_i)$. By Lemma 5.4.5, there exists some other

student $s_z \in M(l)$ such that $s_z \in \rho$, l prefers s_i to s_z, and $s_M(s_z)$ is offered by a lecturer different from l. Now, in the construction of M', s_i is assigned to l (due to the elimination of ρ). At the same time, since $s_z \in \rho$, s_z is no longer assigned to l in M'. Thus, each time a new student is assigned to some lecturer l_k in M' as a result of eliminating ρ, then l_k simultaneously loses a student in $M'(l_k)$. Therefore, $|M(l_k)| = |M'(l_k)|$. Hence, no lecturer is oversubscribed in M'. Since every student is assigned to exactly one project, and no project or lecturer is oversubscribed, it follows that M' is a valid matching.

Now, suppose that M' is not stable. Then there exists a blocking pair (s_i, p_j) in M'. By the construction of M', if s_i is assigned in M', then s_i must also be assigned in M. Let $M(s_i)$ be p_a and let $M'(s_i)$ be p_b. Then, there are three possible conditions on student s_i:

(S1): s_i is unassigned in both M and M';

(S2): s_i is assigned in both M and M', and s_i prefers p_j to both p_a and p_b;

(S3): s_i is assigned in both M and M', s_i prefers p_a to p_j, and prefers p_j to p_b.

Also, there are four possible conditions on the project p_j and the lecturer l_k that offers p_j:

(P1): both p_j and l_k are undersubscribed in M';

(P2): p_j is full in M' and l_k prefers s_i to the worst student in $M'(p_j)$;

(P3): p_j is undersubscribed in M', l_k is full in M', and $s_i \in M'(l_k)$;

(P4): p_j is undersubscribed in M', l_k is full in M', and l_k prefers s_i to the worst student in $M'(l_k)$.

Cases (S1 & P1) or (S2 & P1): We claim that, based on condition (P1), both p_j and l_k are undersubscribed in M. By the construction of M', every lecturer is assigned at least as many students in M' as in M, that is, $|M(l_k)| = |M'(l_k)|$; thus, if l_k is undersubscribed in M', then l_k is undersubscribed in M as well. Similarly, if p_j is undersubscribed in M', then p_j is undersubscribed in M, since by construction, $|M(p_j)| \leq |M'(p_j)|$. If s_i is unassigned in M or prefers p_j to $M(s_i)$, the pair (s_i, p_j) blocks M, contradicting the stability of M. Hence these cases do not hold.

Case (S3 & P1): Following a similar argument as in Cases (S1 & P1) and (S2 & P1), it follows that both p_j and l_k are undersubscribed in M. Since $s_i \in \rho$, s_i prefers p_a to p_j, and prefers p_j to p_b, then by Corollary 5.4.1, (s_i, p_j) is not a stable pair. Hence, this case is impossible.

Cases (S1 & P2) or (S2 & P2): We claim that, based on condition (P2), either l_k prefers s_i to the worst student in $M(p_j)$ if p_j is full in M, or l_k prefers s_i to the worst student in $M(l_k)$ if p_j is undersubscribed in M. To show this, either (a), (b), or (c) holds by the construction of M':

(a) $M(p_j) = M'(p_j)$, that is, p_j has the same set of students in both M and M'. Consequently, p_j is full in M and l_k prefers s_i to the worst student in $M(p_j)$;

(b) $M(p_j) \neq M'(p_j)$, p_j is full in M, and there exists some student $s \in M'(p_j)$ who l_k prefers to the worst student in $M(p_j)$. This implies that l_k prefers s_i to the worst student in $M(p_j)$, since l_k prefers s_i to the worst student in $M'(p_j)$.

(c) $M(p_j) \neq M'(p_j)$, p_j is undersubscribed in M and there exists some student in $s \in M'(l_k)$ who l_k prefers to the worst student in $M(l_k)$. This implies that l_k prefers s_i to the worst student in $M(l_k)$, since l_k prefers s_i to the worst student in $M'(p_j)$.

Hence, our claim holds. We now consider the possible status of s_i in M, that is, s_i is either unassigned in both M and M' or prefers p_j to both p_a and p_b. Given that l_k prefers s_i to the worst student in $M(p_j)$ when p_j is full in M, and similarly prefers s_i to the worst student in $M(l_k)$ when p_j is undersubscribed in M, it follows that the pair (s_i, p_j) blocks M, a contradiction.

Case (S3 & P2): In this case, s_i prefers p_a to p_j and prefers p_j to p_b. By applying a similar argument as in Cases (S1 & P2) and (S2 & P2), we conclude that either l_k prefers s_i to the worst student in $M(p_j)$ if p_j is full in M, or l_k prefers s_i to the worst student in $M(l_k)$ if p_j is undersubscribed in M. First, if p_j is full in M, and l_k prefers s_i to the worst student in $M(p_j)$, it follows directly from the definition of $s_M(s_i)$ that p_j should be a valid $next_M(s_i)$. Consequently, we should have $M'(s_i) = p_j$, yielding a contradiction. Similarly, if p_j is undersubscribed in M and l_k prefers s_i to the worst student in $M(l_k)$, then by the definition of $s_M(s_i)$, p_j must be a valid $next_M(s_i)$, which implies $M'(s_i) = p_j$, another contradiction. Therefore, this blocking pair cannot occur in M'.

Cases (S1 & P3) or (S2 & P3): We claim that, based on condition (P3), p_j is undersubscribed in M, l_k is full in M, and either $s_i \in M(l_k)$ or l_k prefers s_i to the worst student in $M(l_k)$. To show this, either (a) or (b) holds by construction of M':

(a) $M(l_k) = M'(l_k)$, that is, l_k has the same set of students in both M and M'. This implies that p_j is undersubscribed in M, l_k is full in M, and $s_i \in M(l_k)$.

(b) $M(l_k) \neq M'(l_k)$, and there exists some student $s \in M'(l_k)$ such that l_k prefers s to the worst student in $M(l_k)$. First, since p_j is undersubscribed in M', it follows that p_j is also undersubscribed in M since $|M(p_j)| \leq |M'(p_j)|$. Also, by the construction of M', $|M(l_k)| = |M'(l_k)|$. Therefore, l_k is full in M. Now, since l_k prefers s_i to the worst student in $M'(l_k)$ and prefers some student in $s \in M'(l_k)$ to the worst student in $M(l_k)$, it follows that l_k prefers s_i to the worst student in $M(l_k)$.

Therefore, our claim holds: either $s \in M(l_k)$ or l_k prefers s_i to the worst student in $M(l_k)$. We now consider the possible status of s_i in M, that is, s_i is either unassigned in both M and M', or prefers p_j to both p_a and p_b. In this case, since p_j is undersubscribed in M and either $s_i \in M(l_k)$ or l_k prefers s_i to the worst student in $M(l_k)$, it follows that (s_i, p_j) blocks M, a contradiction.

Case (S3 & P3): In this case, s_i is assigned in both M and M', s_i prefers p_a to p_j and prefers p_j to p_b. Clearly, s_i is assigned to different projects in M and M'. By applying a similar argument

as in Cases (S1 & P3) and (S2 & P3), based on condition (P3), it follows that either (a) or (b) holds by construction of M':

(a) $M(l_k) = M'(l_k)$. Consequently, p_j is undersubscribed in M, l_k is full in M, and $s_i \in M(l_k)$. By condition P3, $s_i \in M'(l_k)$, which means that l_k offers p_b. However, by construction of M', if s_i becomes assigned to a different project offered by l_k then l_k simultaneously loses a student in $M(l_k)$. Thus, $M(l_k) \neq M'(l_k)$, a contradiction. Hence, case (a) cannot occur.

(b) $M(l_k) \neq M'(l_k)$, and there exists some student $s \in M'(l_k)$ such that l_k prefers s to the worst student in $M(l_k)$. First, since p_j is undersubscribed in M', it follows that p_j is also undersubscribed in M since $|M(p_j)| \leq |M'(p_j)|$. Also, by the construction of M', $|M(l_k)| = |M'(l_k)|$. Therefore, l_k is full in M. Now, since l_k prefers s_i to the worst student in $M'(l_k)$ and prefers some student in $s \in M'(l_k)$ to the worst student in $M(l_k)$, it follows that l_k prefers s_i to the worst student in $M(l_k)$.

Since p_j is undersubscribed in M and l_k prefers s_i to the worst student in $M(l_k)$, it follows from the definition of $s_M(s_i)$ that p_j must be a valid $next_M(s_i)$, that is, $M'(s_i)$ should be p_j. This leads to a contradiction.

Cases (S1 & P4) or (S2 & P4): Based on condition (P4), it follows that p_j is undersubscribed in M, l_k is full in M, and l_k prefers s_i to the worst student assigned in $M(l_k)$. Specifically, if $M(l_k) = M'(l_k)$, then we have that p_j is undersubscribed in M, l_k is full in M, and l_k prefers s_i to the worst student in $M(l_k)$. Alternatively, if $M(l_k) \neq M'(l_k)$, then there exists some student $s \in M'(l_k)$ such that l_k prefers s to the worst student in $M(l_k)$, which implies that l_k also prefers s_i to the worst student in $M(l_k)$. Hence our claim holds.

We now consider the possible status of s_i in M, that is, s_i is either unassigned in both M and M', or prefers p_j to both p_a and p_b. In this case, since p_j is undersubscribed in M and l_k prefers s_i to the worst student in $M(l_k)$, it follows that (s_i, p_j) blocks M, a contradiction.

Case (S3 & P4): In this case, s_i prefers p_a to p_j and prefers p_j to p_b. By applying a similar argument as in Cases (S1 & P4) and (S2 & P4), we conclude that p_j is undersubscribed in M, l_k is full in M, and l_k prefers s_i to the worst student in $M(l_k)$. Now since p_j is undersubscribed in M and l_k prefers s_i to the worst student in $M(l_k)$, it follows from the definition of $s_M(s_i)$ that p_j must be a valid $next_M(s_i)$, that is, $M'(s_i)$ should be p_j. This leads to a contradiction.

We have now considered all possible conditions for the pair (s_i, p_j) in M', each resulting in a contradiction. Hence, M' is stable. Since every student in ρ receives a less preferred project in M' compared to M, and all other students retain the same projects that they had in M, it follows that M dominates M', that is, M dominates M/ρ. This completes the proof.

□

5.4.3 Meta-rotations and stable matchings

In this section, we further highlight the relationship between meta-rotations and stable matchings of any given instance. We show that every stable matching in a given SPA-S instance can be obtained by eliminating a specific set of meta-rotations starting from the student-optimal stable matching. This connection leads naturally to the definition of the meta-rotation poset in the next section. In Lemma 5.4.7, we show that if ρ is exposed in some stable matching M, and a student $s \in \rho$ prefers M to M', then every student in ρ prefers M to M'. This result is established using Lemmas 5.4.8 to 5.4.10. Moreover, if M dominates M', then either M' is the stable matching obtained by eliminating ρ from M, that is, $M' = M/\rho$, or M/ρ dominates M'.

A key consequence of Lemmas 5.4.6 and 5.4.7 is that it provides a systematic way to construct all stable matchings in a given instance, starting from the student-optimal matching. By successively eliminating an exposed meta-rotation, each step produces a new stable matching in which the students involved in the eliminated meta-rotation are assigned to projects they prefer less to their project in the previous matching. In this way, every stable matching can be reached through a sequence of such eliminations.

Lemma 5.4.7. *Let M and M' be two stable matchings in a given* SPA-S *instance, and let ρ be a meta-rotation exposed in M. Suppose there exists a student $s_i \in \rho$ who prefers M to M'. Then every student $s \in \rho$ prefers M to M'. Moreover, if M dominates M', then either M' is the stable matching obtained by eliminating ρ from M, that is, $M' = M/\rho$, or M/ρ dominates M'.*

Let M and M' be two stable matchings in I, and let ρ be a meta-rotation exposed in M. Suppose there exists a student $s_i \in \rho$ who prefers M to M'. Clearly, $M(s_i) \neq M'(s_i)$, and $s_M(s_i)$ exists. Moreover, s_i prefers $M(s_i)$ to $s_M(s_i)$. By Lemma 5.4.3, there are no projects between $M(s_i)$ and $s_M(s_i)$ that form a stable pair with s_i. Therefore, either $s_M(s_i) = M'(s_i)$, or s_i prefers $s_M(s_i)$ to $M'(s_i)$. Let $p_j = s_M(s_i)$ where l_k offers p_j. By Definition 5.2.2, there exists a student $next_M(s_i)$ in ρ, which we denote by s_z. Since $s_z \in \rho$, $s_M(s_z)$ exists, and s_z prefers $M(s_z)$ to $s_M(s_z)$. By the definition of $next_M(s_i)$ (see Definition 5.2.1), there are two possible conditions on p_j:

(i) p_j is full in M, and s_z is the worst student in $M(p_j)$, or

(ii) p_j is undersubscribed in M, l_k is full in M, and s_z is the worst student in $M(l_k)$.

In both cases (i) and (ii), l_k prefers s_i to s_z.

To prove Lemma 5.4.7, it is enough to show that s_z also prefers M to M'. Once this is established, the same argument can be extended to all other students in ρ. We prove Lemma 5.4.7 using Lemmas 5.4.8 to 5.4.10. Lemma 5.4.8 covers the case where $s_M(s_i) = M'(s_i)$, while Lemmas 5.4.9 and 5.4.10 address the case where s_i prefers $s_M(s_i)$ to $M'(s_i)$. In both Lemmas 5.4.9 and 5.4.10, we first show that s_z is assigned to different projects in M and M', i.e., $M(s_z) \neq M'(s_z)$, and then prove, by contradiction, that s_z prefers M to M'.

Lemma 5.4.8. *Let ρ be an exposed meta-rotation in M, and suppose there exists a student $s_i \in \rho$ who prefers M to M' and $s_M(s_i) = M'(s_i)$. If s_i prefers M to M', then s_z prefers M to M'.*

Proof. Let $s_i \in \rho$ be some student who prefers M to M', and suppose that $s_M(s_i) = M'(s_i)$. This implies that M' is the stable matching obtained by eliminating ρ from M. Moreover, by Lemma 5.4.6, M dominates M'. Recall that $p_j = s_M(s_i)$; thus, $s_i \in M'(p_j) \setminus M(p_j)$. Since s_i is assigned to p_j in M', it follows from Lemma 5.3.1 that, regardless of whether p_j is full or undersubscribed in M, the worst student in $M(p_j)$ or $M(l_k)$, denoted s_z, must be assigned to a different project in M and M'. In particular, $s_z \in M(p_j) \setminus M'(p_j)$. Moreover, since M dominates M', it follows that s_z prefers M to M'. This completes the proof. □

Lemma 5.4.9. *Let ρ be an exposed meta-rotation in M, where $(s_i, p_j) \in \rho$ and s_i prefers p_j to $M'(s_i)$. If p_j is full in M and s_z is the worst student in $M(p_j)$, then s_z prefers M to M'.*

Proof. Let M be a stable matching in which ρ is exposed, and suppose that some student $s_i \in \rho$ prefers M to M'. Let $s_z \in \rho$ be the worst student in $M(p_j)$. We note that l_k prefers s_i to s_z. First suppose for a contradiction that $M(s_z) = M'(s_z)$. Then, regardless of whether p_j is full or undersubscribed in M', the pair (s_i, p_j) blocks M', since s_i prefers p_j to $M'(s_i)$, and l_k prefers s_i to some student in $M'(p_j)$ (namely s_z). This contradicts the stability of M'. Hence, $M(s_z) \neq M'(s_z)$. Now, suppose for a contradiction that s_z prefers M' to M, that is, s_z prefers $M'(s_z)$ to p_j. We consider cases (A) and (B), depending on whether p_j is full or undersubscribed in M'.

(A): p_j is full in M'. Since p_j is also full in M, there exists some student $s_a \in M'(p_j) \setminus M(p_j)$. By Lemma 5.4.1, since s_z prefers $M'(s_z)$ to p_j, l_k prefers s_z to each student in $M'(p_j) \setminus M(p_j)$, so l_k prefers s_z to s_a. Additionally, since s_i prefers p_j to $M'(s_i)$ and p_j is full in M', l_k prefers each student in $M'(p_j)$ to s_i, implying l_k prefers s_a to s_i. Since l_k prefers s_z to s_a, and prefers s_a to s_i, it follows that l_k prefers s_z to s_i. However, by definition of $s_M(s_i)$, l_k prefers s_i to s_z, which yields a contradiction. Therefore, our claim holds and s_z prefers M to M'.

(B): p_j is undersubscribed in M'. By Lemma 5.4.1, since s_z prefers $M'(s_z)$ to p_j, l_k prefers s_z to each student in $M'(l_k) \setminus M(l_k)$. Moreover, if $s_z \in S_k(M, M')$, then by Lemma 5.4.2, there exists at least one student in $M(l_k) \setminus M'(l_k)$ who l_k prefers to s_z, or we have $s_z \in M(l_k) \setminus M'(l_k)$ itself. Consequently, it follows that there also exists a student in $M'(l_k) \setminus M(l_k)$. Let s_b denote the worst student in $M'(l_k) \setminus M(l_k)$. Then l_k prefers s_z to s_b. Since s_i prefers p_j to $M'(s_i)$, and p_j is undersubscribed in M', l_k prefers each student in $M'(l_k)$ (including s_b) to s_i. Since l_k prefers s_z to s_b, and prefers s_b to s_i, it follows that l_k prefers s_z to s_i; This again contradicts the assumption that l_k prefers s_i to s_z (by definition of $next_M(s_i)$). Hence, s_z prefers M to M', and our claim holds. □

Lemma 5.4.10. *Let ρ be an exposed meta-rotation in M, where $(s_i, p_j) \in \rho$ and s_i prefers p_j to $M'(s_i)$. If p_j is undersubscribed in M and s_z is the worst student in $M(l_k)$, then s_z prefers M to M'.*

Proof. Let M be a stable matching in which ρ is exposed, and suppose that some student $s_i \in \rho$ prefers M to M'. Let $s_z \in \rho$ be the worst student in $M(l_k)$. Note that, by definition of $s_M(s_i)$, l_k prefers s_i to s_z. We first show, in case (A), that s_z is assigned to different lecturers in M and M'. We then show, in case (B), that s_z prefers M to M'.

(A): Suppose for a contradiction that $s_z \in M(l_k) \cap M'(l_k)$. We consider subcases (A1) and (A2) depending on whether p_j is full or undersubscribed in M'.

(A1): p_j is full in M'. Since p_j is undersubscribed in M, there exists a student $s_a \in M'(p_j) \setminus M(p_j)$. Since s_i prefers p_j to $M'(s_i)$ and p_j is full in M', it follows that l_k prefers each student in $M'(p_j)$ to s_i. Therefore, l_k prefers s_a to s_i. If s_a prefers p_j to $M(s_a)$, then since p_j is undersubscribed in M, l_k prefers each student in $M(l_k)$ to s_a. In particular, l_k prefers s_z to s_a, since $s_z \in M(l_k)$. Furthermore, since l_k prefers s_z to s_a, and prefers s_a to s_i, it follows that l_k prefers s_z to s_i; this contradicts the fact that l_k prefers s_i to s_z. Therefore, s_a prefers $M(s_a)$ to p_j. Moreover, by Lemma 5.4.1, since p_j is undersubscribed in M, l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$.

Now, since $|M'(p_j)| > |M(p_j)|$ and $|M(l_k)| = |M'(l_k)|$, there exists some project $p_b \in P_k$ such that $|M(p_b)| > |M'(p_b)|$. This implies there exists a student $s_b \in M(p_b) \setminus M'(p_b)$, and p_b is undersubscribed in M'. Moreover, l_k prefers s_b to s_z, since $s_b \in M(l_k)$ and s_z is the worst student in $M(l_k)$. If s_b prefers p_b to $M'(s_b)$, then since p_b is undersubscribed in M', l_k prefers each student in $M'(l_k)$ to s_b. In particular, l_k prefers s_z (who is also in $M'(l_k)$) to s_b, contradicting the earlier fact that l_k prefers s_b to s_z. Therefore, s_b prefers $M'(s_b)$ to p_b. By Lemma 5.4.1 (applied with M and M' swapped), since p_b is undersubscribed in M', l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$.

We now show that the combination of conditions where s_a prefers M to M' and l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$, together with the conditions where s_b prefers M' to M and l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$, leads to a contradiction.

Suppose $s_a \in M'(l_k) \setminus M(l_k)$. Then l_k prefers s_b to s_a, since l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$. Next, suppose $s_a \in S_k(M, M')$. By Lemma 5.4.2, since s_a prefers M to M', then there exists some student $s_r \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s_r to s_a. Given that l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$, it follows that l_k prefers s_b to s_r, and thus l_k prefers s_b to s_a.

A similar argument applies to s_b. Suppose $s_b \in M(l_k) \setminus M'(l_k)$. Then l_k prefers s_a to s_b, since l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$. On the other hand, suppose $s_b \in S_k(M, M')$. By Lemma 5.4.2 (applied with M and M' swapped), there exists a student $s_r \in M(l_k) \setminus M'(l_k)$ such that l_k prefers s_r to s_b. Moreover, since l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$, it follows that l_k prefers s_a to s_r, and thus l_k prefers s_a to s_b. This yields a contradiction since l_k cannot simultaneously prefer s_b to s_a and s_a to s_b. Therefore, the conditions under which s_a prefers M to M', while s_b prefers M' to M, result in a contradiction on the preferences of l_k. Hence, $s_z \in M(l_k) \setminus M'(l_k)$, and this completes the proof for (A1).

(A2): p_j is undersubscribed in M'. Since s_i prefers p_j to $M'(s_i)$, it follows that l_k prefers each student in $M'(l_k)$ to s_i. If $s_z \in M'(l_k)$, then l_k prefers s_z to s_i, which directly contradicts the assumption that l_k prefers s_i to s_z. Hence, $s_z \in M(l_k) \setminus M'(l_k)$.

We now show in case (B) that s_z prefers M to M', given that $M(s_z) \neq M'(s_z)$.

(B): Suppose for a contradiction that s_z prefers M' to M. Again, we consider subcases (B1) and (B2) depending on whether p_j is full or undersubscribed in M'.

(B1): p_j is full in M'. Similar to case (A1), we show that we can identify a student in $M'(l_k) \setminus M(l_k)$ who prefers M to M', and a student in $M(l_k) \setminus M'(l_k)$ who prefers M' to M, which yields a contradiction based on l_k's preferences.

Since $|M'(p_j)| > |M(p_j)|$, there exists a student $s_a \in M'(p_j) \setminus M(p_j)$. Given that s_i prefers p_j to $M'(s_i)$ and p_j is full in M', it follows that l_k prefers s_a to s_i. We also know that l_k prefers s_i to s_z, with $s_z \in M(l_k)$. Therefore, l_k prefers s_a to s_z. Now, if s_a prefers M' to M, then p_j is undersubscribed in M, and l_k would the worst student in $M(l_k)$ (namely s_z) to s_a, which yields a contradiction to the fact that l_k prefers s_a to s_z. Thus, s_a prefers M to M'. In particular, this implies that s_a prefers $M(s_a)$ to p_j, p_j is undersubscribed in M, and by Lemma 5.4.2, l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$.

Recall that $s_z \in M(l_k) \setminus M'(l_k)$ and prefers M' to M. Let $M(s_z)$ be p_z, where $p_z \in P_k$. Let $s_{z'}$ be the worst student in $M'(l_k)$. Since s_z prefers $M'(s_z)$ to p_z, whether p_z is full or undersubscribed in M', it follows that l_k prefers s_z to the worst student in $M'(l_k)$. Therefore l_k prefers s_z to $s_{z'}$.

Now since $|M'(p_j)| > |M(p_j)|$ and $|M(l_k)| = |M'(l_k)|$, there exists a project $p_b \in P_k$ such that $|M(p_b)| > |M'(p_b)|$. This implies that there exists a student $s_b \in M(p_b) \setminus M'(p_b)$, and p_b is undersubscribed in M'. Moreover, l_k prefers s_b to s_z, since $s_b \in M(l_k)$ and s_z is the worst student in $M(l_k)$. If s_b prefers p_b to $M'(s_b)$, then, because p_b is undersubscribed in M', it follows that l_k prefers each student in $M'(l_k)$ to s_b. In particular, l_k prefers $s_{z'}$, the worst student in $M'(l_k)$, to s_b. Additionally, since l_k prefers s_z to $s_{z'}$, it follows that l_k prefers s_z to s_b. However, this contradicts the fact that s_z is the worst student in $M(l_k)$, since it implies that l_k prefers s_z to another student s_b who is also assigned to $M(l_k)$. Therefore, we conclude that s_b prefers $M'(s_b)$ to p_b. By Lemma 5.4.1 (applied with M and M' swapped), since p_b is undersubscribed in M', it follows that l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$.

We now show that combining the conditions where s_a prefers M to M' and l_k prefers s_a to every student in $M(l_k) \setminus M'(l_k)$, together with the conditions where s_b prefers M' to M and l_k prefers s_b to every student in $M'(l_k) \setminus M(l_k)$, leads to a contradiction.

First suppose $s_a \in M'(l_k) \setminus M(l_k)$. Then l_k prefers s_b to s_a, since l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$. Next, suppose $s_a \in S_k(M, M')$ where s_a prefers M to M'. By Lemma 5.4.2,

there exists a student $s_r \in M'(l_k) \setminus M(l_k)$ such that l_k prefers s_r to s_a. Since l_k prefers s_b to each student in $M'(l_k) \setminus M(l_k)$, it follows that l_k prefers s_b to s_r, and thus l_k prefers s_b to s_a.

A similar argument applies to s_b. Suppose $s_b \in M(l_k) \setminus M'(l_k)$. Then l_k prefers s_a to s_b, since l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$. On the other hand, suppose $s_b \in S_k(M, M')$ where s_b prefers M' to M. By Lemma 5.4.2, there exists a student $s_r \in M(l_k) \setminus M'(l_k)$ such that l_k prefers s_r to s_b. Moreover, since l_k prefers s_a to each student in $M(l_k) \setminus M'(l_k)$, it follows that l_k prefers s_a to s_r, and thus l_k prefers s_a to s_b. In both cases, we reach a contradiction, since l_k cannot simultaneously prefer s_b to s_a and s_a to s_b. Therefore, s_z prefers M to M', and this completes the proof.

(B2): p_j is undersubscribed in M'. Since $s_z \in M(l_k) \setminus M'(l_k)$, there exists some student $s_{z'} \in M'(l_k) \setminus M(l_k)$. Since s_i prefers p_j to $M'(s_i)$ and p_j is undersubscribed in M', it follows that l_k prefers each student in $M'(l_k)$ to s_i. In particular, l_k prefers $s_{z'}$ to s_i. Recall that s_z prefers M' to M; let $p_z = M(s_z)$. Whether p_z is full or undersubscribed in M', it follows from Lemma 5.4.1 that l_k prefers s_z to each student in $M'(l_k) \setminus M(l_k)$. In particular, l_k prefers s_z to $s_{z'}$. Combining these observations, we have that l_k prefers s_z to $s_{z'}$, and $s_{z'}$ to s_i, which implies that l_k prefers s_z to s_i. This contradicts the assumption that l_k prefers s_i to s_z. Hence, we conclude that s_z prefers M to M'. Therefore, s_z prefers M to M', and this completes the proof for case (B2).

Thus, in both cases (B1) and (B2), s_z prefers M to M'. This completes the proof. □

The arguments in Lemmas 5.4.9 and 5.4.10 can be extended to every student in ρ, since by Definitions 5.2.1 and 5.2.2, each student in ρ has a valid next student who is also in ρ. Therefore, if $s_i \in \rho$ prefers M to M', then every student $s \in \rho$ also prefers M to M'.

Now, suppose that M dominates M'. By Lemma 5.4.3, for each student $s_i \in \rho$, there is no stable pair that lies between their assigned projects in M and M/ρ. Hence, it follows that M/ρ either dominates M' or is equal to M', since only the students in ρ have different projects in M and M/ρ. Moreover, each of these students prefers M to M', with the possibility that $M/\rho = M'$. This completes the proof of Lemma 5.4.7. In addition, this lemma immediately implies Corollary 5.4.3.

Corollary 5.4.3. *Let $\rho = \{(s_0, p_0), (s_1, p_1), \ldots, (s_{r-1}, p_{r-1})\}$ be a meta-rotation of I. If there exists a stable matching M' such that, for some pair $(s_a, p_a) \in \rho$, student s_a prefers p_a to their project in M', then for every $t \in \{0, \ldots, r-1\}$, student s_t prefers p_t to $M'(s_t)$.*

5.4.3.1 Pruning step

We describe a pruning procedure that constructs a reduced instance $\hat{I}$ from a given SPA-S instance I. First, we apply the student-oriented algorithm to I, which computes the student-optimal stable

matching M_S and removes certain non-stable pairs[2] that cannot be part of any stable matching. We then run the lecturer-oriented algorithm on the resulting instance to compute the lecturer-optimal stable matching M_L, thereby eliminating additional non-stable pairs. The final reduced instance $\hat{I}$ is the instance obtained after running both algorithms.

5.4.3.2 Finding a target stable matching

We now show that any target stable matching M_T in a given instance can be obtained from the student-optimal stable matching by successively exposing and eliminating a sequence of meta-rotations.

Consider a SPA-S instance I and a target stable matching M_T. We start by pruning the instance to obtain $\hat{I}$. Let M denote the student-optimal stable matching in $\hat{I}$. If $M = M_T$, then we are done. Otherwise, if $M \neq M_T$, then there exists a student s such that $M(s) \neq M_T(s)$. Moreover, M dominates M_T in the first step, since the student-optimal stable matching dominates all other stable matchings in I. Therefore, s prefers M to M_T. By Lemma 5.4.4, it follows that there is at least one exposed meta-rotation in M. We identify the meta-rotation ρ that begins at s, and eliminate it to obtain a new stable matching M/ρ, which is guaranteed to be stable by Lemma 5.4.6. Moreover, by Lemma 5.4.7, we have that either $M/\rho = M_T$ or M/ρ dominates M_T (since M dominates M_T). Let $M^* = M/\rho$. If $M^* = M_T$, then we have reached the target matching. Otherwise, since $M^* \neq M_T$, there again exists a student s such that $M^*(s) \neq M_T(s)$. We repeat this process: identify the meta-rotation starting at s, eliminate it, and continue until we reach M_T.

Lemma 5.4.11. *For every stable matching M_T, there exists a set A_T of meta-rotations such that eliminating the meta-rotations in A_T from the student-optimal stable matching yields M_T.*

Proof. Let M denote the student-optimal stable matching in I and let M_T be a target stable matching.

Case 1: $M = M_T$. In this case no eliminations are required to obtain M_T. Setting $A_T = \varnothing$ satisfies the statement of the lemma. Hence the claim holds.

Case 2: $M \neq M_T$. Then there exists a student s such that $M(s) \neq M_T(s)$. By Lemma 5.4.4, there is at least one exposed meta-rotation in M. Let ρ_1 be the exposed meta-rotation beginning at s. Eliminating ρ_1 yields the stable matching M/ρ_1 (Lemma 5.4.6). Furthermore, by Lemma 5.4.7, either $M/\rho_1 = M_T$ or M/ρ_1 differs from M_T only on students who have not yet reached their partners in M_T. If $M/\rho_1 = M_T$, then let $A_T = \{\rho_1\}$ and the proof is complete. Otherwise, set $M^{(1)} = M/\rho_1$. Since $M^{(1)} \neq M_T$, there again exists a student whose assigned project in $M^{(1)}$ differs from that in M_T. By Lemma 5.4.4, an exposed meta-rotation ρ_2 exists in $M^{(1)}$, and eliminating ρ_2 yields the stable matching $M^{(1)}/\rho_2$, which again either equals M_T or differs from it only on students who have a different project in M_T.

[2]A stable pair is one that appears in some stable matching admitted by the instance.

We continue this process. At step k, we obtain the stable matching

$$M^{(k)} = M/\rho_1/\rho_2/\ldots/\rho_k.$$

If $M^{(k)} = M_T$, the process stops. Otherwise, by Lemma 5.4.4 there exists an exposed meta-rotation ρ_{k+1} in $M^{(k)}$, and we eliminate it and continue. Thus, as long as the current matching differs from M_T, an exposed meta-rotation exists, so we may continue eliminating meta-rotations until we arrive at M_T. Let

$$A_T = \{\rho_1, \rho_2, \ldots, \rho_k\}$$

denote the set of meta-rotations eliminated in this sequence. By construction,

$$M_T = M/\rho_1/\rho_2/\cdots/\rho_k,$$

so eliminating the meta-rotations in A_T from the student-optimal stable matching produces M_T. This completes the proof. □

5.4.3.3 Example: Finding all exposed meta-rotations in a SPA-S instance

In this section, we illustrate how to identify all exposed meta-rotations and describe the transitions between stable matchings using the SPA-S instance I_1, shown in Figure 5.1. We begin by constructing the reduced instance corresponding to I_1, following the steps outlined in Section 5.4.3.1.

Now consider instance I_1. From Table 5.1, we observe that M_7 is the lecturer-optimal stable matching for I_1. In M_7, student s_1 is assigned to project p_4, which is the worst project they are assigned to in any stable matching. Consequently, we remove all projects that are less preferred than p_4 from s_1's preference list. Here, project p_3 is deleted from s_1's list. Continuing this pruning process for all students yields the reduced instance for instance I_1, which is presented in Figure 5.5.

s_1: p_1 p_2 p_4	l_1: s_7 s_9 s_3 s_4 s_1 s_2 s_6 s_8	p_1, p_2, p_5, p_6
s_2: p_1 p_4 p_3	l_2: s_6 s_1 s_2 s_5 s_3 s_4 s_7 s_8 s_9	p_3, p_4, p_7, p_8
s_3: p_3 p_1 p_2		
s_4: p_3 p_2 p_1		
s_5: p_4 p_3		
s_6: p_5 p_2 p_7		
s_7: p_7 p_3 p_6		
s_8: p_6 p_8	**Project capacities:** $c_1 = c_3 = 2$; $\forall j \in \{2,4,5,6,7,8\}$, $c_j = 1$	
s_9: p_8 p_2	**Lecturer capacities:** $d_1 = 4$, $d_2 = 5$	

Figure 5.5: Reduced preference list for I_1

Table 5.2 shows, for each student s_i in M_1, the next project p (denoted $s_{M_1}(s_i)$) and the student $\mathrm{next}_{M_1}(s_i)$, defined as either the worst student in $M_1(p)$ if p is full in M, or the worst student in

$M_1(l_k)$ if p is undersubscribed in M. As an illustration, consider s_1: p_2 is the first project after p_1 such that p_2 is undersubscribed in M_1 and l_1 (who offers p_1) prefers s_1 to the worst student in $M_1(l_1)$, namely s_8. Consequently, $\mathrm{next}_{M_1}(s_1) = s_8$. The remaining entries can be verified in a similar manner. We observe that the meta-rotation $\rho_1 = \{(s_8, p_6), (s_9, p_8)\}$ is the only exposed meta-rotation in M_1. Moreover, s_8 is the worst student in p_6 and $\mathrm{next}_{M_1}(s_8) = s_9$. Likewise, s_9 is the worst student in p_8, and $\mathrm{next}_{M_1}(s_9) = s_8$. Eliminating ρ_1 from M_1 gives M_2, that is, $M_1/\rho_1 = M_2$.

(s_i, p_j)	(s_1, p_1)	(s_2, p_1)	(s_3, p_3)	(s_4, p_3)	(s_5, p_4)	(s_6, p_5)	(s_7, p_7)	(s_8, p_6)	(s_9, p_8)
$s_{M_1}(s_i)$	p_2	p_4	p_1	p_2	p_3	p_2	p_6	p_8	p_2
$next_{M_1}(s_i)$	s_8	s_5	s_2	s_8	s_4	s_8	s_8	s_9	s_8

Table 5.2: $s_{M_1}(s_i)$ and $next_{M_1}(s_i)$ for each student s_i in M_1

Similarly, Table 5.3 shows $s_{M_2}(s_i)$ and $\mathrm{next}_{M_2}(s_i)$ for each student s_i in M_2. In M_2, there are two exposed meta-rotations namely $\rho_2 = \{(s_6, p_5), (s_7, p_7)\}$ and $\rho_3 = \{(s_2, p_1), (s_5, p_4), (s_4, p_3)\}$. $M_2/\rho_2 = M_3$ and $M_2/\rho_3 = M_4$.

(s_i, p_j)	(s_1, p_1)	(s_2, p_1)	(s_3, p_3)	(s_4, p_3)	(s_5, p_4)	(s_6, p_5)	(s_7, p_7)	(s_8, p_8)	(s_9, p_2)
$s_{M_2}(s_i)$	p_4	p_4	p_1	p_1	p_3	p_7	p_6	$-$	$-$
$next_{M_2}(s_i)$	s_5	s_5	s_2	s_2	s_4	s_7	s_6	$-$	$-$

Table 5.3: $s_{M_2}(s_i)$ and $next_{M_2}(s_i)$ for each student s_i in M_2

Let M_3 be the next stable matching obtained by eliminating ρ_2 from M_2. Table 5.4 shows $s_{M_3}(s_i)$ and $\mathrm{next}_{M_3}(s_i)$ for each student s_i in M_3. In M_3, there is one exposed meta-rotation namely $\rho_3 = \{(s_2, p_1), (s_5, p_4), (s_4, p_3)\}$. Also, $M_3/\rho_3 = M_5$.

(s_i, p_j)	(s_1, p_1)	(s_2, p_1)	(s_3, p_3)	(s_4, p_3)	(s_5, p_4)	(s_6, p_7)	(s_7, p_6)	(s_8, p_8)	(s_9, p_2)
$s_{M_3}(s_i)$	p_4	p_4	p_1	p_1	p_3	$-$	$-$	$-$	$-$
$next_{M_3}(s_i)$	s_5	s_5	s_2	s_2	s_4	$-$	$-$	$-$	$-$

Table 5.4: $s_{M_3}(s_i)$ and $next_{M_3}(s_i)$ for each student s_i in M_3

Table 5.5 shows $s_{M_5}(s_i)$ and $\mathrm{next}_{M_5}(s_i)$ for each student s_i in M_5. Clearly, the meta-rotation $\rho_4 = \{(s_1, p_1), (s_2, p_4), (s_3, p_3)\}$ is exposed in M_5, and $M_5/\rho_4 = M_7$.

(s_i, p_j)	(s_1, p_1)	(s_2, p_4)	(s_3, p_3)	(s_4, p_1)	(s_5, p_3)	(s_6, p_7)	(s_7, p_6)	(s_8, p_8)	(s_9, p_2)
$s_{M_5}(s_i)$	p_4	p_3	p_1	$-$	$-$	$-$	$-$	$-$	$-$
$next_{M_5}(s_i)$	s_2	s_3	s_1	$-$	$-$	$-$	$-$	$-$	$-$

Table 5.5: $s_{M_5}(s_i)$ and $next_{M_5}(s_i)$ for each student s_i in M_5

We have identified a total of four meta-rotations in instance I_1: ρ_1, ρ_2, ρ_3, and ρ_4, each of which is exposed in at least one stable matching of I_1. We also observe that a meta-rotation can be exposed in multiple stable matchings, and that a single stable matching may contain more than one exposed meta-rotation. For example, the meta-rotation $\rho_2 = \{(s_6, p_5), (s_7, p_7)\}$ is exposed in M_2, M_4, and M_6. Furthermore, the stable matching M_2 contains both ρ_2 and ρ_3 as exposed meta-rotations.

5.5 Meta-rotation poset

In this section, we show that for any SPA-S instance I, we can define a partial order on its set of meta-rotations, forming a partially ordered set (poset), such that each stable matching corresponds to a unique closed subset of this poset.

Given a SPA-S instance I, let $\mathcal{M}$ denote the set of stable matchings in I, and let R be the set of meta-rotations that are exposed in some stable matching in $\mathcal{M}$. For any two meta-rotations $\rho_1, \rho_2 \in R$, we define a relation $\prec$ such that $\rho_1 \prec \rho_2$ if every stable matching in which ρ_2 is exposed can be obtained only after ρ_1 has been eliminated, and there is no other meta-rotation $\rho' \in R \setminus \{\rho_1, \rho_2\}$ such that $\rho_1 \prec \rho' \prec \rho_2$. In this case, we say that ρ_1 is an *immediate predecessor* of ρ_2.

Definition 5.5.1 (Meta-rotation poset)**.** Let R be the set of meta-rotations in a SPA-S instance I, and let $\prec$ be the immediate predecessor relation on R. We define a relation $\leq$ on R such that $\rho_1 \leq \rho_2$ if and only if either $\rho_1 = \rho_2$, or there exists a finite sequence of meta-rotations $\rho_1 \prec \rho_u \prec \cdots \prec \rho_v \prec \rho_2$. The pair $(R, \leq)$ is called the *meta-rotation poset* for instance I.

Proposition 5.5.1. *Let R be the set of meta-rotations in a given SPA-S instance I, and let $\leq$ be the relation on R defined as above. Then $(R, \leq)$ is a partially ordered set.*

Proof. We will show that the relation $\leq$ on R is (i) reflexive, (ii) antisymmetric, and (iii) transitive.

(i) **Reflexivity:** Let $\rho \in R$. By definition, every element is related to itself. Hence, $\rho \leq \rho$, and $\leq$ is reflexive.

(ii) **Antisymmetry:** Suppose there exist $\rho_1, \rho_2 \in R$ such that $\rho_1 \leq \rho_2$ and $\rho_2 \leq \rho_1$. We claim that $\rho_1 = \rho_2$. Suppose, for contradiction, that $\rho_1 \neq \rho_2$. By the definition of $\leq$, there exists a sequence of meta-rotation eliminations $\rho_1 \prec \rho_u \prec \cdots \prec \rho_2$, and another sequence $\rho_2 \prec \rho_v \prec \cdots \prec \rho_1$. Now, consider any stable matching in which ρ_1 is exposed. From the second sequence, we conclude that ρ_2 must have been eliminated before ρ_1 can be exposed. But from the first sequence, ρ_1 must be eliminated before ρ_2 can be exposed. Together,

this implies that neither ρ_1 nor ρ_2 can be exposed without the other having already been eliminated — a contradiction. Therefore, our assumption must be false, and we conclude that $\rho_1 = \rho_2$. Hence, $\leq$ is antisymmetric.

(iii) **Transitivity:** Let $\rho_1, \rho_2, \rho_3 \in R$ such that $\rho_1 \leq \rho_2$ and $\rho_2 \leq \rho_3$. We show that $\rho_1 \leq \rho_3$. By the definition of $\leq$, either $\rho_1 = \rho_2$ or there exists a finite sequence of meta-rotations $\rho_1 \prec \rho_u \prec \cdots \prec \rho_2$, and similarly, either $\rho_2 = \rho_3$ or there exists a finite sequence $\rho_2 \prec \rho_v \prec \cdots \prec \rho_3$. If $\rho_1 = \rho_2$, then $\rho_1 \leq \rho_3$ follows directly from $\rho_2 \leq \rho_3$. If $\rho_2 = \rho_3$, then $\rho_1 \leq \rho_3$ follows from $\rho_1 \leq \rho_2$.

Otherwise, we can combine the two sequences of $\prec$ relations to obtain:

$$\rho_1 \prec \rho_u \prec \cdots \prec \rho_2 \prec \rho_v \prec \cdots \prec \rho_3,$$

which is itself a finite sequence of meta-rotation eliminations from ρ_1 to ρ_3. Therefore, $\rho_1 \leq \rho_3$ by definition of $\leq$, and so the relation is transitive.

□

It follows that $(R, \leq)$ is a partially ordered set. We refer to $(R, \leq)$ as the *meta-rotation poset* of I. For brevity, we will simply write $\Pi(I)$ to refer to this poset throughout the rest of the chapter. Next, we define the closed subset of $\Pi(I)$.

Definition 5.5.2 (Closed subset). A subset of $\Pi(I)$ is said to be *closed* if, for every ρ in the subset, all $\rho' \in R$ such that $\rho' \leq \rho$ are also contained in the subset.

Finally, to prove our result, we present Lemma 5.5.1, which states that no pair (s_i, p_j) belongs to more than one meta-rotation in I.

Lemma 5.5.1. *Let I be a given* SPA-S *instance. No pair (s_i, p_j) can belong to two different meta-rotations in I.*

Proof. Let I be a given SPA-S instance. Suppose for a contradiction that a pair (s_i, p_j) belongs to two different meta-rotations ρ_1 and ρ_2, i.e. $(s_i, p_j) \in \rho_1 \cap \rho_2$ and $\rho_1 \neq \rho_2$. First suppose, without loss of generality, that $\rho_1 \prec \rho_2$. By definition, ρ_1 must be eliminated before ρ_2 can become exposed. Suppose that ρ_1 is exposed in M. Since $(s_i, p_j) \in \rho_1$, eliminating ρ_1 removes the pair (s_i, p_j) from M/ρ_1. Hence (s_i, p_j) is deleted before ρ_2 is exposed. It follows that (s_i, p_j) cannot subsequently appear in ρ_2 (or in any matching dominated by M), giving a contradiction. The same argument applies if $\rho_2 \prec \rho_1$.

Now suppose that the meta-rotations ρ_1 and ρ_2 are distinct, then there exists at least one pair (s', p') such that $(s', p') \in \rho_1 \setminus \rho_2$. We consider cases (A) and (B), depending on whether ρ_1 and

ρ_2 are exposed in the same stable matching or in different ones.

Case (A): ρ_1 and ρ_2 are both exposed in the same stable matching M. Then, $(s_i, p_j) \in M$. Eliminating ρ_2 from M yields a new stable matching $M^* = M/\rho_2$, where each student in ρ_2 is assigned to a less preferred project. So, s_i prefers p_j to $M^*(s_i)$. Let M_L be the lecturer-optimal stable matching. Then either $M^* = M_L$, or M^* dominates M_L. In either case, it follows that s_i is assigned to different projects in M and M_L. By Corollary 5.4.2, any student who is assigned to different projects in M and M_L is involved in at most one exposed meta-rotation of M. Since $s_i \in \rho_2$, and ρ_2 is exposed in M, it follows that s_i cannot also be in ρ_1, contradicting the assumption that $(s_i, p_j) \in \rho_1 \cap \rho_2$.

Case (B): Suppose ρ_1 and ρ_2 are exposed in different stable matchings. Let M_1 be a stable matching in which ρ_1 is exposed, and let M_2 be a stable matching in which ρ_2 is exposed. Recall that $(s_i, p_j) \in \rho_1 \cap \rho_2$, and $(s', p') \in \rho_1 \setminus \rho_2$. Since ρ_2 is exposed in M_2, it follows that $M_2(s_i) = p_j$. Moreover, s' is assigned in M_2. Suppose that s' prefers p' to $M_2(s')$. Then by Corollary 5.4.3, since both (s_i, p_j) and (s', p') are in ρ_1, then s_i also prefers p_j to $M_2(s_i)$; however, this contradicts the fact that $M_2(s_i) = p_j$. Hence, s' either prefers $M_2(s')$ to p', or $M_2(s') = p'$. Let $M_2(s') = p_x$, and let M^* be the stable matching obtained by eliminating ρ_2 from M_2. We consider subcases (B1) and (B2) depending on whether $(s', p_x) \in \rho_2$.

Case (B1): $(s', p_x) \in \rho_2$. Since $(s', p') \notin \rho_2$, we have that $p_x \neq p'$ and s' prefers p_x to p'. After eliminating ρ_2, s_i is worse off in M^* than in M_2, i.e., s_i prefers p_j to $M^*(s_i)$. Meanwhile, s' either becomes assigned to p' (that is, $M^*(s') = p'$), or s' prefers p_x to $M^*(s')$, and prefers $M^*(s')$ to p'. We note that s' does not prefer p' to $M^*(s')$, since by Lemma 5.4.3, if p' lies between p_x and $M^*(s')$ on the preference list of s', then (s', p') is not a stable pair. This means that (s', p') cannot be in ρ_1. Thus, s' does not prefer p' to $M^*(s')$, while s_i prefers p_j to $M^*(s_i)$. Thus, one student (namely s_i) in ρ_1 prefers their project in ρ_1 to their assignment in M^*, while another student (namely s') does not, contradicting Corollary 5.4.3.

Case (B2): $(s', p_x) \notin \rho_2$. Then s' remains assigned to p_x in M^*, that is, $M^*(s') = p_x$. Recall that either s' prefers p_x to p' or $p_x = p'$. By Corollary 5.4.3, since $(s_i, p_j) \in \rho_1$ and s_i prefers p_j to $M^*(s_i)$ then s' should prefer p' to $M^*(s')$, a contradiction.

Therefore, the assumption that $(s_i, p_j) \in \rho_1 \cap \rho_2$ leads to a contradiction in both cases. Hence, no pair belongs to two different meta-rotations in I. □

By Lemma 5.5.1, each student–project pair occurs in at most one meta-rotation. Since there are n_1 students and n_2 projects, there are at most $n_1 n_2$ such pairs, and therefore at most $O(n_1 n_2)$ meta-rotations. We now present a nice structural relationship between the closed subsets of $\Pi(I)$ and the stable matchings of I.

Theorem 5.5.1. *Let I be a* SPA-S *instance. There is a one-to-one correspondence between the set of stable matchings in I and the closed subsets of the meta-rotation poset $\Pi(I)$ of I.*

Proof. Let I be a given SPA-S instance, and let R denote the set of all meta-rotations in I. First, we show that each closed subset of meta-rotations in $\Pi(I)$ corresponds to exactly one stable matching of I. Let $A \subseteq R$ be a closed subset of $\Pi(I)$. By definition, if a meta-rotation $\rho \in A$, then all predecessors of ρ in $\Pi(I)$ also belong to A. Hence, it is possible to eliminate all meta-rotations in A in some order consistent with the partial order $\leq$, starting from the student-optimal stable matching. By Lemma 5.4.6, each such elimination step results in another stable matching of I, and the final matching obtained after eliminating all meta-rotations in A is stable.

Suppose A_1 and A_2 are two distinct closed subsets of $\Pi(I)$. Since $A_1 \neq A_2$, there exists at least one meta-rotation ρ that belongs to one of the subsets and not the other. Furthermore, since no two meta-rotation contains the same set of student-project pairs by Lemma 5.5.1, we would obtain two different stable matchings of I when we eliminate the meta-rotations in A_1 and A_2. Therefore, eliminating each closed subset results in a unique stable matching.

We now prove the converse: that each stable matching $M \in \mathcal{M}$ corresponds to a unique closed subset of $\Pi(I)$. By Lemma 5.4.11, there exists a set of meta-rotations whose elimination produces M. Let $A \subseteq \Pi(I)$ be the set of meta-rotations eliminated from the student-optimal stable matching M_s to obtain M. This set must be closed; that is, if some meta-rotation $\rho_2 \in A$ and $\rho_1 \leq \rho_2$ in $\Pi(I)$, then ρ_1 must have been eliminated before ρ_2 could be exposed, and hence $\rho_1 \in A$. It follows that A contains all predecessors of its elements and is therefore a closed subset.

Now, consider two different stable matchings $M, M' \in \mathcal{M}$. Then there exists a pair $(s_i, p_j) \in M \setminus M'$. We prove that the sets of eliminated meta-rotations that yield M and M' differ. First, suppose M is the student-optimal matching M_s. In this case, no meta-rotation is eliminated to obtain M, but (s_i, p_j) must have been removed during the construction of M' by eliminating some meta-rotation ρ. Thus, ρ is eliminated in the construction of M', but not M. Hence, the sets of eliminated meta-rotations for M and M' are different.

Now suppose $M \neq M_s$. If (s_i, p_j) does not belong to M_s, then (s_i, p_j) must have been introduced to M by eliminating some meta-rotation ρ. By Lemma 5.5.1, each pair appears in at most one meta-rotation. Therefore, s_i becomes assigned to p_j in M through the elimination of exactly one meta-rotation, namely ρ. On the other hand, there are two possibilities for M'. Either ρ was also eliminated in constructing M'; in that case, since $(s_i, p_j) \notin M'$ but $(s_i, p_j) \in M$, at least one additional meta-rotation must have been eliminated when forming M' in order to remove (s_i, p_j) again. Or ρ was eliminated in constructing M but not in constructing M', in which case (s_i, p_j) never appears in M'. If (s_i, p_j) belongs to M_s, then no meta-rotation involving (s_i, p_j) was

eliminated in the construction of M, but (s_i, p_j) must have been removed in the construction of M' by eliminating some meta-rotation ρ. Hence, the sets of eliminated meta-rotations for M and M' differ.

In all cases, the sets of eliminated meta-rotations for M and M' are different. Thus, each stable matching corresponds to a unique closed subset of $\Pi(I)$. □

5.5.1 Example: constructing the meta-rotation poset

Consider instance I_1 shown in Figure 5.1. Although I_1 admits seven stable matchings (see Table 5.1), it contains only four meta-rotations, denoted $R = \{\rho_1, \rho_2, \rho_3, \rho_4\}$. We begin with the student-optimal stable matching M_1, in which only $\rho_1 = \{(s_8, p_6), (s_9, p_8)\}$ is exposed. Eliminating ρ_1 from M_1 yields the matching M_2, where both $\rho_2 = \{(s_6, p_7), (s_7, p_6)\}$ and $\rho_3 = \{(s_2, p_1), (s_4, p_3), (s_5, p_4)\}$ become exposed. Thus, ρ_1 is an *immediate predecessor* of both ρ_2 and ρ_3. From M_2, we can eliminate either ρ_2 (leading to M_3) or ρ_3 (leading to M_4). From M_4, eliminating ρ_2 leads to M_5, and subsequently, eliminating $\rho_4 = \{(s_1, p_1), (s_2, p_4), (s_3, p_3)\}$ from M_5 gives M_7. Alternatively, ρ_4 may be exposed earlier in M_4 by eliminating only ρ_1 and ρ_3. Therefore, ρ_4 depends on ρ_1 and ρ_3, but not on ρ_2. In this case, ρ_1 is a *predecessor* of ρ_4.

Table 5.6 summarises the meta-rotation eliminations observed between the stable matchings in I_1 and the dependencies required for each meta-rotation to become exposed.

From	To	Eliminated meta-rotation	Depends on
M_1	M_2	ρ_1	—
M_2	M_3	ρ_2	ρ_1
M_2	M_4	ρ_3	ρ_1
M_3	M_5	ρ_3	ρ_1
M_4	M_5	ρ_2	ρ_1
M_4	M_6	ρ_4	ρ_1, ρ_3
M_5	M_7	ρ_4	ρ_1, ρ_3
M_6	M_7	ρ_2	ρ_1

Table 5.6: Meta-rotation eliminations in instance I_1.

Figure 5.6 shows the lattice of stable matchings in I_1, where each directed edge corresponds to a single meta-rotation which when eliminated leads to another stable matching.

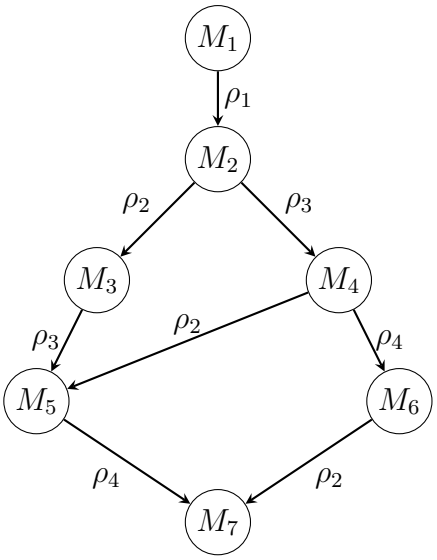

Figure 5.6: Lattice of stable matchings and meta-rotations in I_1.

We now present the meta-rotation poset of I_1. In Figure 5.7, a directed edge from ρ_u to ρ_v indicates that ρ_v can only be exposed once ρ_u has been eliminated. Moreover, each closed subset of $\Pi(I)$ corresponds to a unique stable matching and vice-versa. For example, $\{\rho_1, \rho_3\}$ is closed, while $\{\rho_3\}$ is not, since ρ_1 must be eliminated before ρ_3 becomes exposed. Moreover, $\{\rho_1, \rho_2, \rho_3, \rho_4\}$ is a valid closed subset, as it contains each meta-rotation along with all of its necessary predecessors in the poset. Table 5.7 presents the one-to-one correspondence between the stable matchings in I_1 and the closed subsets of the meta-rotation poset.

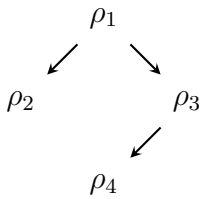

Figure 5.7: Meta-rotation poset $\Pi(I_1)$ for instance I_1.

Stable Matchings of I_1	**Closed Subset of** $\Pi(I_1)$
M_1	$\emptyset$
M_2	$\{\rho_1\}$
M_3	$\{\rho_1, \rho_2\}$
M_4	$\{\rho_1, \rho_3\}$
M_5	$\{\rho_1, \rho_2, \rho_3\}$
M_6	$\{\rho_1, \rho_3, \rho_4\}$
M_7	$\{\rho_1, \rho_2, \rho_3, \rho_4\}$

Table 5.7: Correspondence between stable matchings in I_1 and closed subsets of the meta-rotation poset.

5.6 Conclusions and open problems

In this chapter, we introduced the concept of meta-rotations in SPA-S, by generalising the notion of rotations and meta-rotations from the classical one-to-one and many-to-many settings. Specifically, we proved that there is a one-to-one correspondence between the set of stable matchings in a given instance and the set of closed subsets of the meta-rotation poset $\Pi(M)$. This result provides a compact and structured way to describe and explore the lattice of stable matchings.

The meta-rotation poset also has several algorithmic implications. One immediate consequence is that it provides a method for identifying all stable and non-stable pairs in a given instance and for enumerating the set of stable matchings. However, designing an explicit and efficient algorithm that leverages this structure for SPA-S, similar to the approach of Eirinakis *et al.* [36] for the many-to-many setting, remains an open question. Their algorithm identifies all stable and non-stable pairs in $\mathcal{O}(n^2)$ time and enumerates all stable matchings in $\mathcal{O}(n^2 + n|\mathcal{R}|)$ time, where n is the number of agents and $|\mathcal{R}|$ is the number of stable matchings. Developing a similar algorithm for SPA-S, based on the meta-rotation poset, is a natural and promising direction for future research.

Another direction is to develop a polyhedral characterisation of the set of stable matchings in SPA-S. This would involve defining a set of inequalities whose feasible solutions correspond exactly to the stable matchings in the instance, and proving that the resulting polytope is integral (i.e., each extreme point of the polytope correspond to stable matchings). Establishing such a result could enable new linear programming techniques for solving optimisation problems involving stable matchings in SPA-S. It could also serve as a foundation for showing that the polytopes describing strongly stable and super-stable matchings in SPA-ST are integral, thereby generalising results known for models such as SMTI and HRT. Such a characterisation would also mean that existing results and techniques for linear programming polytopes in simpler matching models can then be generalised in a natural way to SPA-S.

Chapter 6

Conclusions and future directions

In this thesis, we studied the structural and algorithmic aspects of a well-known stable matching problem, the Student-Project Allocation problem (SPA). We focused on two major variants: the Student-Project Allocation problem with lecturer preferences over Projects (SPA-P), and the Student-Project Allocation problem with lecturer preferences over Students (SPA-S). In SPA-P, we presented complexity results for restricted versions of the problem and developed a tractable algorithm for a parameterised variant. In SPA-S, we examined the structural properties of the problem and provided two new characterisations of the set of stable matchings.

In Chapter 3, we focused on determining the boundary between polynomial-time solvability and NP-completeness for the problem of finding a maximum-sized stable matching in various SPA variants. We first examined the extension of SPA-S in which ties are allowed in the preference lists of both students and lecturers. In Section 3.2, we proved that finding a maximum stable matching in SPA-ST, denoted MAX-SPA-ST, remains NP-hard even when the instance involves a single lecturer. We then considered SPA-P with restrictions on the preference lists. In Section 3.3, we showed that finding a maximum stable matching in SPA-P is NP-hard even when student preference lists are derived from a master list of projects, but becomes polynomial-time solvable when each student ranks only projects offered by the same lecturer or when all students have identical preferences. Thereafter, we proved that finding a maximum-size stable matching in a variant called SPA-PUC is fixed-parameter tractable when parameterised by the number of project topics.

Based on these results, a possible direction for future work in the SPA-ST setting is to investigate whether the existing approximation algorithm for MAX-SPA-ST in [27], yields an improved approximation guarantee in the special case where there is only one lecturer. Moreover, given that there are typically more students than lecturers in practical applications, it is reasonable to assume that only lecturers may be permitted to express ties over the students they find acceptable, while each student have strict preferences over a relatively small set of acceptable projects. It would be interesting to investigate how the position and length of ties in lecturer preferences

affect the complexity of MAX-SPA-ST-L1, in a manner similar to the known restrictions on ties presented for MAX-SMTI. A possible direction for future work in the SPA-P setting is to identify a suitable parameter for the general case without assuming uniform capacities, and to determine whether the problem is fixed-parameter tractable with respect to the number of project topics or another appropriate structural parameter, or whether an XP algorithm can be obtained.

In Chapter 4, we proved that the set of stable matchings in SPA-S forms a distributive lattice, with the student-optimal and lecturer-optimal matchings at the top and bottom of the lattice, respectively. A natural future direction is to investigate whether a similar lattice structure holds for the set of strongly stable and super-stable matchings in SPA-ST, as has already been established for SMTI. In terms of designing efficient algorithms, there is an extension of SPA-S in which projects may have both upper and lower quotas, known as SPA-SL$_Q$. In this setting, one seeks a feasible stable matching, meaning a stable matching that also satisfies the upper and lower capacity of each project. However, a feasible solution may not always exist. It remains an open problem to determine whether a polynomial-time algorithm can be devised that either finds a feasible stable matching or correctly reports that none exists. The lattice and structural results presented in this chapter would be instrumental in gaining useful insights to this problem.

In Chapter 5, we characterised the set of stable matchings in SPA-S using meta-rotations, which generalise the notion of rotations from the stable marriage problem. We further developed the meta-rotation poset which compactly encodes the set of all stable matchings in a given SPA-S instance. Specifically, we showed that each stable matching in SPA-S corresponds to a closed subset of a meta-rotation poset. Moreover, the partial order on these subsets captures the dominance relations among stable matchings. This structural characterisation provides a compact and systematic way to represent the set of all stable matchings in a SPA-S instance.

A promising direction for future work is to explore a polyhedral formulation of stable matchings in SPA-S, which, to the best of our knowledge, has not been studied before. A useful starting point is the work of Huang [60], who introduced a system of linear inequalities to describe the stable matching polytope in the Laminar Classified Stable Matching problem (LCSM), and proved that the polytope is integral. We note that LCSM can be viewed as a special case of SPA-S if each classification (representing projects) forms a disjoint partition of the applicants, and there are no lower bounds. In this setting, the applicants correspond to students, and the lecturers correspond to institutes. In particular, if the polytope associated with a SPA-S instance is shown to be integral, then suitable objective functions could be defined to compute target stable matchings that meet different optimality criteria, such as the median stable matching.

Bibliography

[1] Cplex optimization studio. https://www.ibm.com/products/ilog-cplex-optimization-studio. Accessed: 2025-05-16.

[2] Gurobi optimization website. https://www.gurobi.com/. Accessed: 2025-05-30.

[3] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[4] Hernán Abeledo and Yosef Blum. Stable matchings and linear programming. *Linear algebra and its applications*, 245:321–333, 1996.

[5] David J Abraham, Katarína Cechlárová, David F Manlove, and Kurt Mehlhorn. Pareto optimality in house allocation problems. In *International symposium on algorithms and computation*, pages 3–15. Springer, 2004.

[6] David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

[7] David J Abraham, Robert W Irving, and David F Manlove. The student-project allocation problem. In *Algorithms and Computation: 14th International Symposium, ISAAC 2003, Kyoto, Japan, December 15-17, 2003. Proceedings 14*, pages 474–484. Springer, 2003.

[8] David J Abraham, Robert W Irving, and David F Manlove. Two algorithms for the student-project allocation problem. *Journal of discrete algorithms*, 5(1):73–90, 2007.

[9] Deeksha Adil, Sushmita Gupta, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. Parameterized algorithms for stable matching with ties and incomplete lists. *Theoretical Computer Science*, 723:1–10, 2018.

[10] José Alcalde. Implementation of stable solutions to marriage problems. *Journal of Economic Theory*, 69(1):240–254, 1996.

[11] Arif A Anwar and AS Bahaj. Student project allocation using integer programming. *IEEE Transactions on Education*, 46(3):359–367, 2003.

[12] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[13] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. On the structure of combinatorial problems and structure preserving reductions. In *International Colloquium on Automata, Languages, and Programming*, pages 45–60. Springer, 1977.

[14] Vipul Bansal, Aseem Agrawal, and Varun S Malhotra. Polynomial time algorithm for an optimal stable assignment with multiple partners. *Theoretical Computer Science*, 379(3):317–328, 2007.

[15] Mourad Baïou and Michel Balinski. The stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research*, 27(3):485–503, 2002.

[16] Garrett Birkhoff. Rings of sets. *Duke Math. Journal*, 3(1):443–454, 1937.

[17] Péter Biró. The stable matching problem and its generalizations: an algorithmic and game theoretical approach. *Unpublished PhD thesis, BME, Mathematics and Computer Science Doctoral School, Budapest*, 2007.

[18] Péter Biró and Tamás Fleiner. The integral stable allocation problem on graphs. *Discrete Optimization*, 7(1-2):64–73, 2010.

[19] Niclas Boehmer, Klaus Heeger, and Rolf Niedermeier. Deepening the (parameterized) complexity analysis of incremental stable matching problems. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, pages 21–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022.

[20] Richard P Brent. Recent progress and prospects for integer factorisation algorithms. In *International Computing and Combinatorics Conference*, pages 3–22. Springer, 2000.

[21] Son Thanh Cao, Le Quoc Anh, and Hoang Huu Viet. A heuristic repair algorithm for the hospitals/residents problem with ties. In *International Conference on Artificial Intelligence and Soft Computing*, pages 340–352. Springer, 2022.

[22] Son Thanh Cao, Le Van Thanh, and Hoang Huu Viet. Finding maximum weakly stable matchings for hospitals/residents with ties problem via heuristic search. In *Australasian Joint Conference on Artificial Intelligence*, pages 442–454. Springer, 2023.

[23] Katarína Cechlárová and Tamás Fleiner. On a generalization of the stable roommates problem. *ACM Transactions on Algorithms (TALG)*, 1(1):143–156, 2005.

[24] Katarína Cechlárová and Veronika Val'ová. The stable multiple activities problem. Technical Report 1, Mathematical Institute, Slovak Academy of Sciences, 2005.

[25] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[26] Christine Cheng, Eric McDermid, and Ichiro Suzuki. A unified approach to finding good stable matchings in the hospitals/residents setting. *Theoretical Computer Science*, 400(1-3):84–99, 2008.

[27] Frances Cooper and David Manlove. A 3/2-approximation algorithm for the student-project allocation problem. *arXiv preprint arXiv:1804.02731*, 2018.

[28] Vincent P. Crawford and Edward M. Knoer. Job matching with heterogeneous firms and workers. *Econometrica: Journal of the Econometric Society*, 49(2):437–450, 1981.

[29] Gergely Csáji, David Manlove, Iain McBride, and James Trimble. Couples can be tractable: New algorithms and hardness results for the hospitals/residents problem with couples. *arXiv preprint arXiv:2311.00405*, 2023.

[30] Ágnes Cseh, Tobias Friedrich, and Jannik Peters. Pareto optimal and popular house allocation with lower and upper quotas. *arXiv preprint arXiv:2107.03801*, 2021.

[31] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[32] Brian Dean and Rommel Jalasutram. Factor revealing LPs and stable matching with ties and incomplete lists. In *Proceedings of the 3rd International Workshop on Matching Under Preferences*, pages 42–53, 2015.

[33] Brian C. Dean and Sudarshan Munshi. Faster algorithms for stable allocation problems. *Algorithmica*, 58:59–81, 2010.

[34] Maxence Delorme, Sergio García, Jacek Gondzio, Jörg Kalcsics, David Manlove, and William Pettersson. Mathematical models for stable matching problems with ties and incomplete lists. *European Journal of Operational Research*, 277(2):426–441, 2019.

[35] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[36] Pavlos Eirinakis, Dimitrios Magos, Ioannis Mourtos, and Panayiotis Miliotis. Finding all stable pairs and solutions to the many-to-many stable matching problem. *INFORMS Journal on Computing*, 24(2):245–259, 2012.

[37] Ahmed H Abu El-Atta and Mahmoud Ibrahim Moussa. Student project allocation with preference lists over (student, project) pairs. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 1, pages 375–379. IEEE, 2009.

[38] Enrico Maria Fenoaltea, Izat B Baybusinov, Jianyang Zhao, Lei Zhou, and Yi-Cheng Zhang. The stable marriage problem: An interdisciplinary review from the physicist's perspective. *Physics Reports*, 917:1–79, 2021.

[39] Tamás Fleiner. A matroid generalization of the stable matching polytope. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 105–114. Springer, 2001.

[40] Tamás Fleiner. A fixed-point approach to stable matchings and some applications. *Mathematics of Operations research*, 28(1):103–126, 2003.

[41] Tamás Fleiner, András Frank, and Tamás Király. A new approach to bipartite stable matching optimization. *arXiv preprint arXiv:2409.04885*, 2024.

[42] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987.

[43] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[44] David Gale and Marilda Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

[45] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[46] Tomáš Gavenčiak, Dušan Knop, and Martin Koutecký. Integer programming in parameterized complexity: Three miniatures. *arXiv preprint arXiv:1711.02032*, 2017.

[47] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K Brent Venable, and Toby Walsh. Local search approaches in stable matching problems. *Algorithms*, 6(4):591–617, 2013.

[48] Frederik Glitzner and David Manlove. Structural and algorithmic results for stable cycles and partitions in the roommates problem. In *International Symposium on Algorithmic Game Theory*, pages 3–20. Springer, 2024.

[49] Frederik Glitzner and David Manlove. Unsolvability and beyond in many-to-many non-bipartite stable matching. *arXiv preprint arXiv:2505.11456*, 2025.

[50] Yunan Gu, Walid Saad, Mehdi Bennis, Merouane Debbah, and Zhu Han. Matching theory for future wireless networks: Fundamentals and applications. *IEEE Communications Magazine*, 53(5):52–59, 2015.

[51] Sushmita Gupta, Saket Saurabh, and Meirav Zehavi. On treewidth and stable marriage. *arXiv preprint arXiv:1707.05404*, 2017.

[52] Dan Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

[53] Dan Gusfield. The structure of the stable roommate problem: efficient representation and enumeration of all stable assignments. *SIAM Journal on Computing*, 17(4):742–769, 1988.

[54] Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[55] Magnús M Halldórsson, Robert W Irving, Kazuo Iwama, David F Manlove, Shuichi Miyazaki, Yasufumi Morita, and Sandy Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306(1-3):431–447, 2003.

[56] Paul R Harper, Valter de Senna, Israel T Vieira, and Arjan K Shahani. A genetic algorithm for the project assignment problem. *Computers & Operations Research*, 32(5):1255–1265, 2005.

[57] Nguyen Thuy Hoa, Tran Van Hoai, Hoang Huu Viet, et al. Finding max-smti for stable marriage with ties and bounded preference lists. In *2019 International Conference on Advanced Computing and Applications (ACOMP)*, pages 107–111. IEEE, 2019.

[58] Jeffrey Hoffstein. Integer factorization and rsa. In *An introduction to mathematical cryptography*, pages 1–75. Springer, 2008.

[59] Changyong Hu and Vijay K Garg. Characterization of super-stable matchings. In *Algorithms and Data Structures: 17th International Symposium, WADS 2021, Virtual Event, August 9–11, 2021, Proceedings 17*, pages 485–498. Springer, 2021.

[60] Chien-Chung Huang. Classified stable matching. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1235–1253. SIAM, 2010.

[61] Chien-Chung Huang and Telikepalli Kavitha. An improved approximation algorithm for the stable marriage problem with one-sided ties. In *Integer Programming and Combinatorial Optimization: 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings 17*, pages 297–308. Springer, 2014.

[62] Robert W Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, 1985.

[63] Robert W Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48(3):261–272, 1994.

[64] Robert W Irving. Optimal stable marriage. In *Encyclopedia of Algorithms*, pages 606–609. Springer, 2008.

[65] Robert W Irving and Paul Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[66] Robert W Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543, 1987.

[67] Robert W Irving and David F Manlove. Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems. *Journal of Combinatorial Optimization*, 16(3):279–292, 2008.

[68] Robert W Irving, David F Manlove, and Gregg O'Malley. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms*, 7(2):213–219, 2009.

[69] Robert W Irving, David F Manlove, and Sandy Scott. The hospitals/residents problem with ties. In *Scandinavian Workshop on Algorithm Theory*, pages 259–271. Springer, 2000.

[70] Robert W Irving, David F Manlove, and Sandy Scott. Strong stability in the hospitals/residents problem. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 439–450. Springer, 2003.

[71] Robert W Irving, David F Manlove, and Sandy Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008.

[72] Robert W Irving and Sandy Scott. The stable fixtures problem—a many-to-many extension of stable roommates. *Discrete Applied Mathematics*, 155(16):2118–2129, 2007.

[73] Kazuo Iwama, David Manlove, Shuichi Miyazaki, and Yasufumi Morita. Stable marriage with incomplete lists and ties. In Harald Ganzinger, editor, *Automata, Languages and Programming: 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11–15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 443–452, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[74] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. In *Theory and Applications of Models of Computation: 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011. Proceedings 8*, pages 440–451. Springer, 2011.

[75] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. A 25/17-approximation algorithm for the stable marriage problem with one-sided ties. *Algorithmica*, 68(3):758–775, 2014.

[76] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[77] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[78] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[79] Jonathan A Kelner and Daniel A Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 51–60, 2006.

[80] Tamás Király and Júlia Pap. Total dual integrality of rothblum's description of the stable-marriage polyhedron. *Mathematics of Operations Research*, 33(2):283–290, 2008.

[81] Zoltán Király. Better and simpler approximation algorithms for the stable marriage problem. *Algorithmica*, 60(1):3–20, 2011.

[82] Zoltán Király. Linear time local approximation algorithm for maximum stable marriage. *Algorithms*, 6(3):471–484, 2013.

[83] Dušan Knop, Martin Koutecký, and Matthias Mnich. Voting and bribing in single-exponential time. *ACM Transactions on Economics and Computation (TEAC)*, 8(3):1–28, 2020.

[84] Donald Ervin Knuth. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. American Mathematical Soc., 1997.

[85] Donald Ervin Knuth. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. American Mathematical Soc., 1997.

[86] Adam Kunysz. An algorithm for the maximum weight strongly stable matching problem. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2018.

[87] Adam Kunysz, Katarzyna Paluch, and Pratik Ghosal. Characterisation of strongly stable matchings. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 107–119. SIAM, 2016.

[88] Augustine Kwanashie, Robert W Irving, David F Manlove, and Colin TS Sng. Profile-based optimal matchings in the student/project allocation problem. In *Combinatorial Algorithms: 25th International Workshop, IWOCA 2014, Duluth, MN, USA, October 15-17, 2014, Revised Selected Papers 25*, pages 213–225. Springer, 2015.

[89] Augustine Kwanashie and David F Manlove. An integer programming approach to the hospitals/residents problem with ties. In *Operations Research Proceedings 2013: Selected Papers of the International Conference on Operations Research, OR2013, organized by the German Operations Research Society (GOR), the Dutch Society of Operations Research (NGB) and Erasmus University Rotterdam, September 3-6, 2013*, pages 263–269. Springer, 2014.

[90] Somdeb Lahiri. Stable matchings for a generalised marriage problem. Technical report, Nota di Lavoro, 2003.

[91] Chi-Kit Lam and C Gregory Plaxton. A (1+ 1/e)-approximation algorithm for maximum stable matching with one-sided ties and incomplete lists. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2823–2840. SIAM, 2019.

[92] Arjen K Lenstra. Integer factoring. In *Encyclopedia of Cryptography and Security*, pages 290–297. Springer, 2005.

[93] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[94] David Manlove. *Algorithmics of matching under preferences*, volume 2. World Scientific, 2013.

[95] David Manlove, Duncan Milne, and Sofiat Olaosebikan. An integer programming approach to the student-project allocation problem with preferences over projects. In *Combinatorial Optimization: 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11–13, 2018, Revised Selected Papers*, pages 313–325. Springer, 2018.

[96] David Manlove, Duncan Milne, and Sofiat Olaosebikan. Student-project allocation with preferences over projects: Algorithmic and experimental results. *Discrete applied mathematics*, 308:220–234, 2022.

[97] David F Manlove. Stable marriage with ties and unacceptable partners. Technical report, Technical Report TR-1999-29, University of Glasgow, Department of Computing Science, 1999.

[98] David F Manlove. The structure of stable marriage with indifference. *Discrete Applied Mathematics*, 122(1-3):167–181, 2002.

[99] David F Manlove. Hospitals/residents problem. In *Encyclopedia of Algorithms*, pages 390–394. Springer, 2008.

[100] David F Manlove, Robert W Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[101] David F Manlove, Iain McBride, and James Trimble. "almost-stable" matchings in the hospitals/residents problem with couples. *Constraints*, 22:50–72, 2017.

[102] David F Manlove and Gregg O'Malley. Student-project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6(4):553–560, 2008.

[103] David F Manlove and Gregg O'malley. Paired and altruistic kidney donation in the uk: Algorithms and experimentation. *Journal of Experimental Algorithmics (JEA)*, 19:1–21, 2015.

[104] David F Manlove and Colin TS Sng. Popular matchings in the capacitated house allocation problem. In *European Symposium on Algorithms*, pages 492–503. Springer, 2006.

[105] Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[106] Dániel Marx and Ildikó Schlotter. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica*, 58(1):170–187, 2010.

[107] Dániel Marx. The multivariate algorithmics revolution (invited talk, tel aviv 2017), 2017. `https://www.cs.bme.hu/~dmarx/papers/marx-telaviv2017-hardness.pdf`.

[108] Yuki Matsuyama and Shuichi Miyazaki. Hardness of instance generation with optimal solutions for the stable marriage problem. *Journal of Information Processing*, 29:166–173, 2021.

[109] Iain McBride. *Complexity results and integer programming models for hospitals/residents problem variants*. PhD thesis, University of Glasgow, 2015.

[110] Eric McDermid. A 3/2-approximation algorithm for general stable marriage. In *International Colloquium on Automata, Languages, and Programming*, pages 689–700. Springer, 2009.

[111] Eric J McDermid and David F Manlove. Keeping partners together: algorithmic results for the hospitals/residents problem with couples. *Journal of Combinatorial Optimization*, 19:279–303, 2010.

[112] D. G. McVitie and L. B. Wilson. Algorithm 411: Three procedures for the stable marriage problem. *Communications of the ACM*, 14(7):491–492, 1971.

[113] D. G. McVitie and L. B. Wilson. The stable marriage problem. *Communications of the ACM*, 14(7):486–490, 1971.

[114] Kitty Meeks and Baharak Rastegari. Solving hard stable matching problems involving groups of similar agents. *Theoretical Computer Science*, 844:171–194, 2020.

[115] Danny Munera, Daniel Diaz, Salvador Abreu, Francesca Rossi, Vijay Saraswat, and Philippe Codognet. A local search algorithm for smti and its extension to hrt problems. In *3rd International Workshop on Matching Under Preferences*, 2015.

[116] Danny Munera, Daniel Diaz, Salvador Abreu, Francesca Rossi, Vijay Saraswat, and Philippe Codognet. A local search algorithm for smti and its extension to hrt problems. In *3rd International Workshop on Matching Under Preferences*, 2015.

[117] Milind Nasre and Ankit Rawat. Popularity in the generalized hospital residents setting. In Anil Nerode, Igor Kotenko, Pavel Skobelev, and Andrei Yakovlev, editors, *International Computer Science Symposium in Russia*, volume 10304 of *Lecture Notes in Computer Science*, pages 245–259, Cham, 2017. Springer International Publishing.

[118] National Resident Matching Program. Main residency match data and reports. Web document available at `https://www.nrmp.org/match-data-analytics/residency-data-reports/` (Accessed 08 May 2025).

[119] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, 1988.

[120] Uyen T Nguyen and Sang X Tran. Advanced heuristic solution for the hospital-resident matching with ties problem. In *International Conference on Computational Data and Social Networks*, pages 383–394. Springer, 2024.

[121] Sofiat Olaosebikan. *The Student-Project Allocation problem: structure and algorithms*. PhD thesis, University of Glasgow, 2020.

[122] Sofiat Olaosebikan and David Manlove. An algorithm for strong stability in the student-project allocation problem with ties. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 384–399. Springer, 2020.

[123] Sofiat Olaosebikan and David Manlove. Super-stability in the student-project allocation problem with ties. *Journal of Combinatorial Optimization*, 43(5):1203–1239, 2022.

[124] Gregg O'Malley. *Algorithmic aspects of stable matching problems*. PhD thesis, University of Glasgow, 2007.

[125] JP Owen. An evaluation of the medical training application service as experienced by defence medical service medical officers. *BMJ Military Health*, 153(3):175–180, 2007.

[126] Katarzyna Paluch. Faster and simpler approximation of stable matchings. *Algorithms*, 7(2):189–202, 2014.

[127] BS Panda and Sachin. Hardness and approximation results for some variants of stable marriage problem. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 252–264. Springer, 2022.

[128] András Radnai. Approximation algorithms for the stable matching problem. *Eötvös Lorand University*, 2014.

[129] Alvin E Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016, 1984.

[130] Alvin E Roth. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica: Journal of the Econometric Society*, pages 425–427, 1986.

[131] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research*, 18(4):803–828, 1993.

[132] Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3):828–851, 2007.

[133] Alvin E Roth and Marilda Sotomayor. The college admissions problem revisited. *Econometrica: Journal of the Econometric Society*, pages 559–570, 1989.

[134] Alvin E. Roth and Xiaolin Xing. Jumping the gun: Imperfections and institutions related to the timing of market transactions. *The American Economic Review*, 84(4):992–1044, 1994.

[135] Uriel G Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.

[136] KA Santhini, Govind S Sankar, and Meghana Nasre. Optimal matchings with one-sided preferences: fixed and cost-based quotas. *Autonomous Agents and Multi-Agent Systems*, 39(1):1–36, 2025.

[137] Sandy Scott. *A Study Of Stable Marriage Problems With Ties*. PhD thesis, University of Glasgow, 2005.

[138] Boris Spieker. The set of super-stable marriages forms a distributive lattice. *Discrete applied mathematics*, 58(1):79–84, 1995.

[139] Jimmy JM Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12(1):154–178, 1991.

[140] Jimmy JM Tan. Stable matchings and stable partitions. *International Journal of Computer Mathematics*, 39(1-2):11–20, 1991.

[141] Y Teo and Duan Juat Ho. A systematic approach to the implementation of final year project in an electrical engineering undergraduate course. *IEEE transactions on education*, 41(1):25–30, 1998.

[142] Paul Tseng. A simple complexity proof for a polynomial-time linear programming algorithm. *Operations Research Letters*, 8(3):155–159, 1989.

[143] Nguyen Thi Uyen, Giang L Nguyen, Canh V Pham, Tran Xuan Sang, and Hoang Huu Viet. A heuristic algorithm for student-project allocation problem. In *International Conference on Computational Data and Social Networks*, pages 280–291. Springer, 2022.

[144] John H Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

[145] Hoang Huu Viet, Nguyen Thi Uyen, Son Thanh Cao, and Long Giang Nguyen. An efficient two-heuristic algorithm for the student-project allocation with preferences over projects. *Journal of Intelligent & Fuzzy Systems*, pages JIFS–236300, 2024.

[146] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[147] Niklaus Wirth. *Algorithms + Data Structures = Programs*, volume 158. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[148] Hiroki Yanagisawa. *Approximation Algorithms for Stable Marriage Problems*. PhD thesis, Kyoto University, Japan, 2003.

[149] Hongmou Zhang and Jinhua Zhao. Mobility sharing as a preference matching problem. *IEEE Transactions on Intelligent Transportation Systems*, 20(7):2584–2592, 2018.