Efficient Algorithms for Optimal Matching Problems Under Preferences

Augustine Kwanashie

Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

September 2015

Abstract

In this thesis we consider efficient algorithms for matching problems involving preferences, i.e., problems where agents may be required to list other agents that they find acceptable in order of preference. In particular we mainly study the *Stable Marriage problem* (SM), the *Hospitals / Residents problem* (HR) and the *Student / Project Allocation problem* (SPA), and some of their variants. In some of these problems the aim is to find a *stable* matching which is one that admits no *blocking pair*. A blocking pair with respect to a matching is a pair of agents that prefer to be matched to each other than their assigned partners in the matching if any.

We present an *Integer Programming* (IP) model for the *Hospitals / Residents problem with Ties* (HRT) and use it to find a maximum cardinality stable matching. We also present results from an empirical evaluation of our model which show it to be scalable with respect to real-world HRT instance sizes.

Motivated by the observation that not all blocking pairs that exist in theory will lead to a matching being undermined in practice, we investigate a relaxed stability criterion called *social stability* where only pairs of agents with a social relationship have the ability to undermine a matching. This stability concept is studied in instances of the *Stable Marriage problem with Incomplete lists* (SMI) and in instances of HR. We show that, in the SMI and HR contexts, socially stable matchings can be of varying sizes and the problem of finding a maximum socially stable matching (MAX SMISS and MAX HRSS respectively) is NP-hard though approximable within 3/2. Furthermore we give polynomial time algorithms for three special cases of the problem arising from restrictions on the social network graph and the lengths of agents' preference lists. We also consider other optimality criteria with respect to social stability and establish inapproximability bounds for the problems of finding an *egalitarian*, *minimum regret* and *sex equal* socially stable matching in the SM context.

We extend our study of social stability by considering other variants and restrictions of MAX SMISS and MAX HRSS. We present NP-hardness results for MAX SMISS even under certain restrictions on the degree and structure of the social network graph as well as the presence of master lists. Other NP-hardness results presented relate to the problem of determining whether a given man-woman pair belongs to a socially stable

matching and the problem of determining whether a given man (or woman) is part of at least one socially stable matching. We also consider the *Stable Roommates problem with Incomplete lists under Social Stability* (a non-bipartite generalisation of SMI under social stability). We observe that the problem of finding a maximum socially stable matching in this context is also NP-hard. We present efficient algorithms for three special cases of the problem arising from restrictions on the social network graph and the lengths of agents' preference lists. These are the cases where (i) there exists a constant number of acquainted pairs (ii) or a constant number of unacquainted pairs or (iii) each preference list is of length at most 2.

We also present algorithmic results for finding matchings in the SPA context that are optimal with respect to *profile*, which is the vector whose ith component is the number of students assigned to their ith-choice project. We present an efficient algorithm for finding a *greedy maximum matching* in the SPA context — this is a maximum matching whose profile is lexicographically maximum. We then show how to adapt this algorithm to find a *generous maximum matching* — this is a matching whose reverse profile is lexicographically minimum. We demonstrate how this approach can allow additional constraints, such as lecturer lower quotas, to be handled flexibly. We also present results of empirical evaluations carried out on both real world and randomly generated datasets. These results demonstrate the scalability of our algorithms as well as some interesting properties of these profile-based optimality criteria.

Practical applications of SPA motivate the investigation of certain special cases of the problem. For instance, it is often desired that the workload on lecturers is evenly distributed (i.e. load balanced). We enforce this by either adding lower quota constraints on the lecturers (which leads to the potential for infeasible problem instances) or adding a load balancing optimisation criterion. We present efficient algorithms in both cases. Another consideration is the fact that certain projects may require a minimum number of students to become viable. This can be handled by enforcing lower quota constraints on the projects (which also leads to the possibility of infeasible problem instances). A technique of handling this infeasibility is the idea of *closing projects* that do not meet their lower quotas (i.e. leaving such project completely unassigned). We show that the problem of finding a maximum matching subject to project lower quotas where projects can be closed is NP-hard even under severe restrictions on preference lists lengths and project upper and lower quotas. To offset this hardness, we present polynomial time heuristics that find large feasible matchings in practice. We also present IP models for the SPA variants discussed and show results obtained from an empirical evaluation carried out on both real and randomly generated datasets. These results show that our algorithms and heuristics are scalable and provide good matchings with respect to profile-based optimality.

Table of Contents

1 Introduction **1**

2 Literature Review of Matching Problems **5**

2.1 Introduction 5

2.2 The Stable Marriage problem (SM) 7

2.2.1 Introduction 7

2.2.2 Structure of stable matchings in SM 9

2.2.3 Stable Marriage problem with Incomplete lists 12

2.2.4 Stable Marriage problem with Ties 12

2.2.5 Stable Marriage problem with Ties and Incomplete lists 13

2.2.6 Other optimality criteria 14

2.3 The Hospitals/Residents problem (HR) 15

2.3.1 Introduction 15

2.3.2 Hospital/Residents with Ties 16

2.3.3 Cloning HR instances 18

2.4 The Stable Roommates problem (SR) 19

2.4.1 Rotations in the Stable Roommates problem 20

2.4.2 Structure of stable matchings in SR 20

2.4.3 Stable Roommates with Incomplete lists 20

2.5 Locally Stable Matchings 21

2.5.1 Introduction 21

2.5.2 Locally Stable Matchings in the Job Market Context 22

2.5.3 Other results relating to Locally Stable Matchings 24

2.6 The Student/Project Allocation problem 24
2.6.1 Introduction 24
2.6.2 Two-sided preferences and stability 25
2.6.3 One-sided preferences and profile-based optimality 27
2.6.4 Other SPA models and approaches 29
2.7 Integer Programming approaches to matching problems 30

3 An Integer Programming Approach to the Hospitals/Residents Problem with Ties 32

3.1 Introduction 32
3.2 An IP model for MAX HRT 33
3.3 Implementing the model 35
3.3.1 Reducing the model size 35
3.3.2 Improving optimisation performance 37
3.3.3 Testing the implementation 38
3.4 Empirical evaluation 39
3.4.1 Using random instances 40
3.4.2 Using real instances 43
3.5 Open problems 44

4 Socially Stable Matchings in the Hospitals/Residents Problem 46

4.1 Introduction 46
4.2 Preliminary definitions and results 47
4.3 Reduction from HRSS to HR+SN 48
4.4 Hardness of MAX SMISS 50
4.5 Approximating MAX HRSS 51
4.5.1 Approximation results 52
4.5.2 Inapproximability result 57
4.6 Some special cases of HRSS 58
4.6.1 $(2, \infty)$-MAX SMISS 59
4.6.2 HRSS with a constant number of unacquainted pairs 62

4.6.3 HRSS with a constant number of acquainted pairs 63
4.7 Empirical evaluation 67
4.7.1 Introduction 67
4.7.2 Varying instance size 69
4.7.3 Varying the density of the social network 69
4.8 Conclusion 71

5 Further Algorithmic Results on Socially Stable Matchings 73
5.1 Introduction 73
5.2 Fair socially stable matchings 74
5.2.1 Introduction 74
5.2.2 Egalitarian socially stable matchings 75
5.2.3 Minimum regret socially stable matchings 77
5.2.4 Sex-equal socially stable matchings 78
5.3 Further restrictions of COM SMISS 80
5.3.1 Restrictions on the degree of the social network graph 80
5.3.2 Restrictions on the structure of the social network graph 83
5.3.3 Restrictions on the ordering of preference lists 84
5.4 Minimum socially stable matchings 86
5.5 Further hardness results for SMISS 89
5.6 The roommates problem under social stability 90
5.6.1 Introduction 90
5.6.2 MAX SRISS with a constant number of unacquainted pairs 92
5.6.3 MAX SRISS with a constant number of acquainted pairs 92
5.6.4 2-MAX SRISS 95
5.7 Conclusion 98

6 Profile-based optimal matchings in the Student/Project Allocation problem 99
6.1 Introduction 99
6.2 Preliminary definitions 100

6.3 Greedy maximum matchings in SPA . . . 103
6.4 Generous maximum matchings in SPA . . . 116
6.5 Integer Programming models for SPA . . . 117
6.5.1 Introduction . . . 117
6.5.2 Model with exponential coefficients (`Model1`) . . . 117
6.5.3 Model with hierarchical objectives (`Model2`) . . . 118
6.6 Empirical evaluation . . . 120
6.6.1 Introduction . . . 120
6.6.2 Testing for correctness . . . 121
6.6.3 Feasibility analysis of the MCMF approach . . . 123
6.6.4 Real-world data . . . 124
6.6.5 Random data . . . 126
6.6.6 Concluding remarks . . . 130
6.7 Conclusion . . . 131

7 Further Algorithmic Results for SPA and its Variants 133
7.1 Introduction . . . 133
7.2 Lecturer lower quotas and load balancing . . . 134
7.2.1 Introduction . . . 134
7.2.2 Lecturer lower quotas . . . 134
7.2.3 Minimising matching span . . . 139
7.2.4 Minimising span with lower quotas . . . 141
7.3 Project lower quotas . . . 142
7.3.1 Introduction . . . 142
7.3.2 Hardness of MAX SPA-PL . . . 143
7.3.3 Heuristics for MAX SPA-PL . . . 145
7.4 Extending the SPA IP model . . . 148
7.5 Empirical evaluation . . . 149
7.5.1 Introduction . . . 149
7.5.2 Lecturer lower quotas . . . 150
7.5.3 Minimising matching span . . . 151

7.5.4 Evaluating Heuristics for MAX SPA-PL 152
7.6 Conclusion 154

8 Further Experimental Results for SM and SR **156**
8.1 Introduction 156
8.2 Implementation 157
8.2.1 The `AlgBreakmarriage` algorithm 157
8.2.2 The `GetAllStableMatchingsSM` algorithm 159
8.2.3 The `GetAllStableMatchingsSR` algorithm 159
8.3 Empirical Evaluation 160

Bibliography **164**

List of Tables

3.1 Percentage solvable instances (100% for omitted t_d values) 40

3.2 SFAS IP Results 43

4.1 Mean runtime (ms) produced by both algorithms and CPLEX 70

4.2 Mean matching sizes produced by both algorithms and CPLEX 70

6.1 Real-world SPA instances 125

6.2 Real-world SPA results 125

6.3 Mean matching profile and cost 131

7.1 Span of matchings in real-world SPA instances 151

7.2 Mean sizes of matchings produced by various heuristics and CPLEX . . 154

List of Figures

2.1 An instance of the Stable Marriage problem 8
2.2 An instance of the Stable Marriage problem with Ties 13
2.3 An instance of the Stable Marriage problem with Ties 13
2.4 An instance of the Hospitals/Residents problem 16
2.5 An instance of the Stable Roommates problem 19
2.6 An instance of the Stable Roommates problem with Incomplete Lists . 21
2.7 A SPA instance I 28

3.1 `Model1`: A HRT IP model 34
3.2 Mean runtime vs t_d 41
3.3 Median runtime vs t_d 41
3.4 $|M|$ vs t_d for $n_1 = 300$ 41
3.5 Range vs t_d 41
3.6 Mean and median runtime vs instance size 43
3.7 Optimal solution size vs instance size 43

4.1 An SMISS instance (I, G) consisting of a HR instance I and a social network G 48
4.2 $|M_{opt}| = (3/2).|M|$ 58
4.3 A MAX HRSS IP model 68
4.4 Mean matching size vs n_1 69
4.5 Mean runtime vs n_1 69
4.6 Mean matching size vs d_G 70
4.7 Mean runtime vs d_G 70

5.1 No man-optimal socially stable matching in SMISS instance (I, G) . . . 74
5.2 Rural Hospitals Theorem fails in SMISS instance (I, G) 74
5.3 Preference lists in the constructed instance of COM SMISS. 81
5.4 Pictorial representation of the preference lists. 81
5.5 Preference lists in the constructed instance of COM SMISS-2ML. 85
5.6 A cycle of type-2 blocking pairs 96

6.1 A SPA instance I 103
6.2 Some types of compound path in X' 107
6.3 `Model1`: IP model for finding a greedy maximum matching given a SPA instance 118
6.4 `Model2`: IP model for finding a maximum matching given a SPA instance 118
6.5 Mean runtime vs n_1 122
6.6 Mean runtime vs n_1 122
6.7 Mean runtime vs R 122
6.8 MCMF feasibility results 124
6.9 Mean matching degree vs n_1 127
6.10 Mean runtime vs n_1 127
6.11 Mean matching cost vs n_1 127
6.12 Mean matching cost vs R 128
6.13 Mean matching size vs R 128
6.14 Mean matching degree vs R 129
6.15 Mean matching cost vs popularity 130
6.16 Mean matching size vs popularity 130
6.17 Mean matching degree vs popularity 130

7.1 A SPA-L instance I 135
7.2 % solvability vs $L_{\mathcal{L}}/C_{\mathcal{L}}$ 150
7.3 Minimum span vs n_1 152
7.4 Mean quality vs n_1 152
7.5 Mean matching size vs n_1 153
7.6 Mean runtime vs n_1 153

7.7 Mean matching degree vs n_1 153

8.1 Visualising the rotation poset of an SM instance 157
8.2 Mean $|\mathcal{S}|$ vs $n \ln n$ (SM) 162
8.3 Mean $|\mathcal{S}|$ vs n (SR) 162
8.4 Mean stable pairs vs n (SM) 162
8.5 Mean stable pairs vs n (SR) 162
8.6 Mean cost vs n (SM) 163
8.7 Mean cost vs n (SR) 163
8.8 Mean regret vs n (SM) 164
8.9 Mean regret vs n (SR) 164
8.10 Mean number of rotations vs n (SM) 164
8.11 Mean number of rotations vs n (SR) 164

List of Algorithms

2.1 Alg-GS 9
2.2 Alg-EGS 9
3.1 Hospitals-offer 36
3.2 Residents-apply 36
3.3 Generate-max-hrt 39
3.4 Choose 39
4.1 Approx-smiss 55
4.2 Mod-exgs 55
4.3 $(2, \infty)$-Max-smiss-alg 60
4.4 BuildGraph 60
5.1 2-max sriss-alg 97
6.1 Greedy-max-spa 109
6.2 Get-max-aug (method for Greedy-max-spa) 112
6.3 Hierarchy-spa-ip 119
7.1 Greedy-max-spa-l 138
7.2 Alg-min-span 141
7.3 Heuristic-cps 146
7.4 Heuristic-ops 148

Abbreviations

CHAT	Capacitated House Allocation problem with Ties
HA	House Allocation problem
HR	Hospitals/Residents problem
HR+SN	Hospitals/Residents problem with Social Network
HR-LQ	Hospitals/Residents problem with Lower Quotas
HROST	Hospitals/Residents problem with One-Sided Ties
HRSS	Hospitals/Residents problem under Social Social Stability
HRT	Hospitals/Residents problem with Ties
IP	Integer Programming
MCMF	Minimum Cost Maximum Flow
NRMP	National Resident Matching Program
SFAS	Scottish Foundation Allocation Scheme
SM	Stable Marriage problem
SMI	Stable Marriage problem with Incomplete lists
SMISS	Stable Marriage problem with Incomplete lists under Social Stability
SMISS-1ML	Stable Marriage problem with Incomplete lists under Social Stability and 1 Master Lists
SMISS-2ML	Stable Marriage problem with Incomplete lists under Social Stability and 2 Master Lists
SMSS	Stable Marriage problem under Social Stability
SMT	Stable Marriage problem with Ties
SMTI	Stable Marriage problem with Ties and Incomplete lists
SPA	Student/Project Allocation problem
SPA-L	Student/Project Allocation problem with Lecturer lower quotas
SPA-P	Student/Project Allocation problem with lecturer preferences over Projects
SPA PL	Student/Project Allocation problem with Project Lower quotas
SPA-S	Student/Project Allocation problem with lecturer preferences over Students
SPA-(S, P)	Student/Project Allocation problem with lecturer preferences over (Student, Project) pairs
SR	Stable Roommates problem

SRF	Stable Roommates problem with Free edges
SRI	Stable Roommates problem with Incomplete preference lists
SRISS	Stable Roommates problem with Incomplete preference lists under Social Stability
SS PAIR-SMISS	Socially Stable pair in SMISS

Acknowledgments

I would like to thank my parents Mike and Helen Kwanashie who sacrificed so much to fund my graduate studies and my supervisor David for all his guidance and feedback. Many thanks to Lisa and Judith for all their love and support. Thanks to Gethin Norman, Rob Irving, Iain McBride and other colleagues and friends for their help throughout my PhD. Thanks also to Daniel Paulusma and Kitty Meeks for their very detailed feedback on the submitted version.

Declaration

This thesis is submitted in accordance with the rules 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. Theorem 4.6.11 was due to Zoltán Király, Theorem 7.3.1 was due to David Manlove and Theorem 7.3.3 was due to Ágnes Cseh. Otherwise all the results contained herein are claimed as original.

Publications

1. A. Kwanashie and D. F. Manlove. An Integer Programming approach to the Hospitals/Residents problem with Ties. In *Proceedings of OR 2013: the International Conference on Operations Research*, pages 263–269. Springer, 2014. (This paper is based on Chapter 3.)

2. G. Askalidis, N. Immorlica, A. Kwanashie, D. F. Manlove, and E. Pountourakis. Socially Stable matchings in the Hospitals/Residents problem. In *Proceedings of* WADS 2013: the 13th Algorithms and Data Structures Symposium, volume 8037 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2013. (This paper is based on Chapter 4.)

3. A. Kwanashie, R. W. Irving, D. F. Manlove and C. T. S. Sng. Profile-based optimal matchings in the Student/Project Allocation problem. In *Proceedings of* IWOCA 2014: 25th International Workshop on Combinatorial Algorithms (to appear) (This paper is based on Chapter 6. A longer version of this paper has been submitted to a special issue of the *European Journal of Combinatorics* devoted to selected papers from IWOCA 2014).

4. U. Krähmer, A. Kwanashie, D. F. Manlove and M. Zelvyte. Student/Project Allocation problem using Network Flow and Integer Programming. (in preparation) (This paper is based on Chapter 7).

Chapter 1

Introduction

Matching problems, in their most general form, involve assigning agents from one set to those of another. In these problems some agents are required to have ordinal preferences over a subset of the others. These problems find their applications in various centralised matching schemes around the world. Matching problems were first introduced in the seminal paper by David Gale and Lloyd Shapley titled "College Admissions and the Stability of Marriage" [29]. Here, they gave a formal definition of the *Stable Marriage Problem* (SM) which involves forming a one-one pairing of a set of n men to a set of n women where no man and woman prefer each other to their assigned partners. They generalised SM to a many-one problem called the *Hospitals/Residents Problem* (HR) and gave polynomial time algorithms for solving both SM and HR. They also defined a non-bipartite generalisation of SM called the *Stable Roommates Problem* (SR). Since then research in the area has grown rapidly with a wide range of matching problems and algorithms being described in the literature [38, 85].

In some matching schemes (like school choice and labour markets) agents are allowed to negotiate with each other directly and form pairs individually. However these *free-for-all* markets have been shown to be prone to various problems in practice when considering the number and satisfaction of the agents involved [101, 107, 106]. A more sustainable approach is to adopt *centralised matching schemes* where a central authority accepts agents' preferences and computes a matching based on predefined criteria.

As computational speed and network performance have increased over the years and electronic forms of communication have become ubiquitous, the advantages of setting up centralised matching schemes have became more obvious. This trend also means that centralised matching schemes are involving an increasing number of participants and a varying set of feasibility and optimality criteria thus leading to a host of research problems. Due to the typical size and intricacy of these matching schemes, manual or

brute-force techniques for solving these matching problems have become infeasible in practice and so algorithms are needed to solve these problems efficiently. Since execution time is a very important requirement in practice, a lot of the research effort has gone into developing *efficient* (polynomial-time) algorithms for solving these matching problems. These factors have led to a lot of research activity in the area.

Some of the applications of matching problems include the National Resident Matching Program (NRMP) where medical residents are allocated to residency programs in the US [101] and other similar schemes like the Scottish Foundation Allocation Scheme (SFAS) [52]. Other matching schemes can be found in the context of allocating kidney donors to patients such as the UK's National Living Donor Kidney Sharing Schemes [16]. Here, patients with chronic kidney disease obtain compatible donors by swapping their own willing but incompatible donors. Assignment of pupils to schools [1, 2], conference papers to reviewers [33] and students to final year projects [8, 87, 68, 9] are also examples of the applications of matching problems.

In matching problems where agents have preferences, often a key requirement is for a matching to be stable. However certain scenarios exist where stability is either too restrictive or irrelevant. For example in two-sided matching problems where preferences exist on one side only, stability cannot always be enforced. Also in scenarios where agents are unable to abandon their partners (maybe due to existing rules or limited information among agents), stability may be considered to be too strong a constraint. In these scenarios a weaker form of stability may be more appropriate. This argument is further strengthened by the observation that enforcing stability often tends to limit the size of the matchings obtained. Indeed in some matching problems a stable matching may be half the size of a maximum matching. Since maximising the number of assigned agents is usually desired, it makes sense that weaker forms of stability that allow for larger matchings should be investigated.

In this thesis we investigate two such weakened stability criteria, namely *weak stability* (in the context of HR instances with ties) and *social stability* (in instances of HR and other problems). In the case of weak stability, agents' preference lists are allowed to contain ties. Thus a pair can only block a matching if both agents *strictly* prefer each other. In the case of social stability, we rule out certain agent pairs from blocking any matching thus allowing socially stable matchings to be potentially larger than stable matchings. We demonstrate how adopting these stability definitions can improve the size of the matchings obtained while discouraging agents from abandoning their assigned partners. We also investigate a class of problems (in the domain of allocating students to projects) where stability is no longer appropriate and matchings are considered valid only based on agents' capacity constraints. This is because, in these problems, preference lists are only provided by the students and so blocking pairs

cannot be formed.

Finding a feasible solution (for example, a stable matching) is often the first step in solving some matching problems. The set of feasible solutions may be (exponentially) large with each solution having varying measures with respect to certain desirable properties. For example in the variant of HR where agents are allowed to be indifferent between others on their preference lists, the set of solutions contains matchings of varying sizes. In these cases we seek to find an *optimal* solution/matching (subject to feasibility) based on predefined optimality criteria. One obvious optimality criterion is the *size* of a matching. Others are based on the *profile*, *cost* and *degree* of a matching as well as various notions of "fairness" of a matching. For example, in the case of assigning students to projects, the profile of a matching indicates the number of students assigned their 1st, 2nd, 3rd, etc project in the matching, the cost of a matching is the sum of the ranks of the assigned students in the matching (where the rank indicates the position of her assigned project on her preference list) and the degree of a matching is the worse rank of any student in the matching. In this thesis we investigate a number of these optimality criteria which are mainly motivated by practical applications. We provide a wide range of algorithmic and experimental results relating to these optimal matching problems.

The remainder of the thesis is structured as follows. Chapter 2 gives a review of the literature on matching problems involving preferences with particular focus on the variants that will feature in this thesis. In Chapter 3, an *Integer Programming* (IP) model for the MAX HRT problem is presented along with implementation details and results from empirical evaluations. These evaluations involved measuring the runtime and matching size obtained as we varied the size and the density of ties of randomly-generated HRT instances. Chapters 4 and 5 focus on a relaxed stability criterion called *social stability* where only pairs of agents with a social relationship have the ability to undermine a matching. In Chapter 4 we show that, in the SM and HR contexts, socially stable matchings can be of varying sizes and the problem of finding a maximum socially stable matching (MAX SMISS and MAX HRSS respectively) is NP-hard though approximable within 3/2. Furthermore we give polynomial time algorithms for three special cases of the problem. We also present results from an empirical evaluation of the approximation algorithm described in the chapter. In Chapter 5, we focus on other optimality criteria relating to fairness in the context of socially stable matchings and prove some inapproximability results for problems involving computing three types of fair socially stable matchings. Further algorithmic results for computing socially stable matchings in variants of SM and SR are also presented.

In Chapters 6 and 7 we drop the stability criterion altogether and focus on profile-based optimality criteria in the *Student / Project Allocation problem* (SPA) which involves

assigning students to projects offered by lecturers on the basis of student preferences over projects, and project and lecturer capacities. In Chapter 6 we provide efficient algorithms for finding profile-based optimal matchings given SPA instances. We present IP models for SPA and its variants. We also provide results from an empirical evaluation of the techniques described in the chapter. These evaluations measure the size, cost and degree of the matchings as well as the time taken while varying the size, preference list lengths and other properties of randomly-generated SPA instances. Chapter 7 considers other optimality and feasibility criteria. These include minimising a load-balancing objective, and introducing lower quotas on projects and lecturers. We provide IP models, heuristics and efficient algorithms for the SPA problems with these extensions. We also perform an empirical evaluation were we measure the size, degree and other properties of the matchings obtained as we vary the size of randomly-generated SPA instances.

Finally in Chapter 8 we present further experimental results for SM and SR. In particular we focus on the algorithms for finding the set of all stable matchings given an SM or SR instance. These experiments involve measuring the quantity, egalitarian cost and minimum regret of stable matchings obtained from randomly generated SM and SR instances of varying sizes. In egalitarian stable matchings we seek to optimise the satisfaction of all agents involved while in minimum regret stable matchings we seek to minimise the worse-off agent in the matching. We also describe the algorithms implemented as well as a tool for visualising the structures they produce (i.e., rotation posets, rotation digraphs and Hasse diagrams).

Chapter 2

Literature Review of Matching Problems

2.1 Introduction

Matching problems generally involve the assignment of a set (or sets) of agents to one another. Some agents are required to list a subset of other agents that they find acceptable in order of preference (thus forming *preference lists*) [38, 85]. Two agents form an *acceptable pair* if they both appear on each other's preference list. All agents involved are subject to *capacity* constraints which specify the maximum number of assignments they can have. A *matching* is a set of acceptable pairs in which all agents involved obey their capacity constraints.

Two criteria commonly used in the literature for classifying matching problems are the number of disjoint sets of agents involved and the presence of preference lists. Thus the majority of matching problems in the literature can be grouped into the following classes:

1. *Bipartite matching problems with two-sided preferences.* These problems involve two disjoint sets of agents with the agents of each set providing preference lists containing agents of the other. Examples arise from the assignment of doctors to hospitals [101, 52] and pupils to schools [1, 2].

2. *Bipartite matching problems with one-sided preferences.* These problems also involve two disjoint sets of agents but only agents in one set provide preference lists containing agents of the other. Applications arise in assigning students to campus housing [98] and conference papers to reviewers [33].

3. *Non-bipartite matching problems with preferences.* These problems involve only one set of agents with each agent's preference list containing other agents in the same set. Such problems occur in the context of kidney exchange schemes [104, 103, 6] and pairing players in chess tournaments [78].

In this thesis we focus mainly on problems that fall under groups 1 and 3 above. We also investigate a group of matching problems that can be regarded as extensions of groups 1 and 2. In particular we consider problems involving the assignment of students to projects offered by lecturers. These problems involve three disjoint sets of agents with preference lists appearing in one or two of these sets. However, like in groups 1 and 2, the objective is to find bipartite matchings.

In each of the three groups identified above, matching problems may vary from one another due to a number of properties. These include:

1. *The capacity constraints involved.* Agent capacity constraints may enable one-to-one, many-to-one or many-to-many matchings to be found.

2. *The lengths and structure of preference lists.* Agents may be allowed to omit others whom they find unacceptable from their preference lists thus leading to *incomplete* preference lists [30, 61] and *bounded-length* preference lists [61]. They may also be allowed to be indifferent between two or more agents on their preference lists thus leading to *ties* [64, 62, 58]. Cases may also exists where all the preference lists from a given set of agents are derived from a uniform global ranking of agents (also known as *a master list*) [63].

3. *The solution/optimality criteria employed.* A matching is *stable* if no two agents would rather be assigned to each other than to their partners if any. In many practical applications of matching problems, stability is a desired property of the matchings generated [101]. However other feasibility and optimality criteria may be employed either in conjunction with or in place of stability. Although a large proportion of the literature in this area focuses on stability [38], research into these alternative/complementary solution criteria continues to grow [85, chapters 6-8].

All of the matching problems we study in this thesis involve some optimality criteria based on agents' preferences. We seek to develop *efficient algorithms* to solve these problems, that is, algorithms whose time complexity is a polynomial function of the size of the given input. However some of these optimisation problems turn out to be NP-hard (as shown in the literature or in this thesis). In these cases, we may seek to develop *approximation algorithms* where possible. These are algorithms that produce solutions

which are sub-optimal but are guaranteed to lie within some fraction of optimality. We also prove inapproximability bounds for some of these NP-hard matching problems. Other solution techniques for matching problems that may be theoretically inefficient but practically feasible have also been investigated in the literature (and also in this thesis) [114, 38, 108, 102, 109, 10, 105, 6, 113, 70, 20, 89, 17]. These techniques scale well with typical problem instances despite their exponential worst-case running times. They include Linear Programming (LP), Integer Programming (IP) and Constraint Programming (CP).

This chapter surveys the various matching problems involving preferences investigated in this thesis. Firstly, in Section 2.2, the classical *Stable Marriage problem* (SM) is introduced. Its variants, involving ties and/or incomplete preference lists, are also discussed. The many-one generalisation of SM, the *Hospitals/Residents problem* (HR), is also considered along with its variant involving ties, in Section 2.3. The non-bipartite generalisation of SM, the *Stable Roommates problem* (SR) is then discussed in Section 2.4. We also review results relating to variants of SR with incomplete preference lists. In Section 2.5, we review the literature around *social/local* stability and give some motivation for adopting this relaxed stability criterion. In Section 2.6 we discuss the *The Student/Project Allocation problem* (SPA). We review the literature on two-sided preferences in the SPA context and we also discuss alternative optimality criteria in the case of one-sided preferences. Finally we briefly discuss integer programming approaches to matching problems involving preferences in Section 2.7.

2.2 The Stable Marriage problem (SM)

2.2.1 Introduction

The *Stable Marriage problem* (SM) [29] is a bipartite matching problem involving a set of n men and a set of n women. Each person has a strictly-ordered preference list of all n partners of the opposite sex, thus the preference lists are said to be *complete* and the SM instance is said to be of *size* n. A matching in this context contains a set of n man-woman pairs where each man and woman appear exactly once. A man m_i *prefers* woman w_j to w_k if w_j appears before w_k on his preference list. The same definition holds for a woman's preferences over men on her list. We denote the woman assigned to a man m_i in a matching M as $M(m_i)$ and the man assigned to a woman w_j in M as $M(w_j)$. A pair (m_i, w_j) is said to *block* M, or form a *blocking pair* with respect to M, if m_i prefers w_j to $M(m_i)$ and w_j prefers m_i to $M(w_j)$. A matching is *stable* if it admits no blocking pair. In a stable matching no two agents have any

incentive to break up their assignments and become assigned to each other. A pair (m_i, w_j) is said to be a *stable pair* if it exists in some stable matching. Figure 2.1 shows an example of an SM instance I involving 4 men and 4 women. The matching $M = \{(m_1, w_4), (m_2, w_3), (m_3, w_2), (m_4, w_1)\}$ is a stable matching in I.

men's preferences				women's preferences			
$m_1 : w_2$	w_4	w_1	w_3	$w_1 : m_2$	m_1	m_4	m_3
$m_2 : w_3$	w_1	w_4	w_2	$w_2 : m_4$	m_3	m_1	m_2
$m_3 : w_2$	w_3	w_1	w_4	$w_3 : m_1$	m_4	m_3	m_2
$m_4 : w_4$	w_1	w_3	w_2	$w_4 : m_2$	m_1	m_4	m_3

Figure 2.1: An instance of the Stable Marriage problem

The Stable Marriage problem was first introduced by Gale and Shapley in their seminal paper 'College admissions and the stability of marriage' [29]. They showed that, given an SM instance of size n, a stable matching can always be found in polynomial-time, and moreover they also provided an $O(n^2)$ algorithm to find such a matching. The algorithm involves a sequence of marriage proposals made by the members of one sex to the members of the other. Each proposal is followed by either an acceptance or a rejection. Proposals are accepted if the person being proposed to is either unmatched or prefers the proposer to his/her current partner, in which case his/her current partner is rejected in favour of the new proposal. Proposals are rejected if the person being proposed to already has a partner he/she prefers to the proposer. The algorithm can be executed with the men doing the proposing (*man-oriented*) or with the women doing the proposing (*woman-oriented*). Agents of the proposing set start with the most preferred agents on their preference lists and make proposals until they are accepted. If they ever get rejected due to their partner getting a better proposal from someone else, they continue by making a proposal to the next agent on their preference list. The algorithm terminates when all parties involved are matched. Algorithm 2.1 shows the basic man-oriented Gale-Shapley algorithm (`Alg-GS`). Since the total input size of the problem including the preference lists is $\Theta(n^2)$, the algorithm runs in linear time with respect to the input size.

Depending on the way the Gale-Shapley algorithm is executed (i.e., whether the man-oriented or woman-oriented version is run), the stable matching generated is either *man-optimal* or *woman-optimal*. We refer to a man-optimal matching as one in which no man can have a better partner in any other stable matching. A man-optimal stable matching is also a *woman-pessimal* one, meaning that no woman can get a worse partner in any other stable matching. Analogous definitions exists for a woman-optimal and a man-pessimal stable matching. The SM instance I shown in Figure 2.1

Algorithm 2.1 `Alg-GS`

```
1: set each person to be free;
2: while some man m is free do
3:    w = most preferred woman on m's list to which he has not yet proposed;
4:    if w is free then
5:       assign m to w;
6:    else
7:       if w prefers m to her current partner m' then
8:          assign m to w to be engaged and set m' to be free;
9:       else
10:         w rejects m's proposal and remains with m'; {m remains free}
11: the stable matching consists of all n engagements;
```

Algorithm 2.2 `Alg-EGS`

```
1: set each person to be free;
2: while some man m is free do
3:    w = first woman on m's list;
4:    if w is engaged to m' then
5:       set m' to be free;
6:    assign m and w to be engaged;
7:    for each successor m'' of m on w's list do
8:       delete m'' from w's preference list and w from m''s preference list;
9: the stable matching consists of all n engagements;
```

admits the man-optimal stable matching $M_0 = \{(m_1, w_4), (m_2, w_3), (m_3, w_2), (m_4, w_1)\}$ and the woman-optimal stable matching $M_z = \{(m_1, w_4), (m_2, w_1), (m_3, w_2), (m_4, w_3)\}$. In some cases, the man-oriented and woman-oriented versions of the algorithm can produce the same stable matching, i.e., $M_0 = M_z$. In this case the problem instance admits a unique stable matching.

A simple modification to the basic Gale-Shapley algorithm aims to cut down the number of proposals that need to be made by identifying, at each stage in the proposal sequence, certain pairs that cannot be involved in any stable matching. Algorithm 2.2 shows the so-called Extended Gale-Shapley algorithm [38] (`Alg-EGS`). This reduction in the number of proposals is possible because at each proposal stage of, say, the man-oriented version of the algorithm, a woman will automatically accept any proposal that she receives and will remove all men less desirable than her partner from her preference list (and her from theirs). This means that any future proposals from men poorer than her current partner will no longer be possible as she is also removed from their list.

2.2.2 Structure of stable matchings in SM

As stated above, every instance of SM admits a stable matching. Depending on the orientation of the Gale-Shapley algorithm used, the stable matching obtained may be

man-optimal or woman-optimal. In many cases an SM instance will admit more than one stable matching. Although all the stable matchings generated are of the same size and are *complete* (a complete matching is one in which all men and women are matched), they vary with respect to other important properties. For example, stable matchings differ according to the relative satisfaction of the men and women involved. The algorithms described so far only generate the man-optimal or woman-optimal stable matching but none of the other stable matchings the instance may admit. Thus algorithms that generate the set of all stable matchings given an instance of SM have been developed. As we shall state later, these algorithms have exponential running time in the worst case.

For an instance of SM, the elements in the set of all stable matchings form a lattice structure under a dominance relation, with the matching at the top of the lattice being man-optimal and the one at the bottom woman-optimal. We note that an agent a *prefers* a matching M to another matching M' if he/she prefers $M(a)$ to $M'(a)$. If $M(a) = M'(a)$, we say a is *indifferent* between both matchings. The aforementioned dominance relation can be defined in terms of man-oriented or woman-oriented dominance. Considering man-oriented dominance, for a given instance of SM, a matching M is said to *dominate* another matching M' (we denote this as $M \preceq M'$) if each man either prefers M to M' or is indifferent between them. The set of all stable matchings forms a distributive lattice under this dominance relation (Knuth in [76] attributes this result to John Conway). This is a partially ordered set in which every pair of matchings has a unique meet and a unique join and these meet and join operations distribute over each other. This distributive lattice can be represented as a *Hasse diagram* which is a directed graph with the stable matchings as nodes and directed edges running from a node M to another node M' if $M \preceq M'$ and there is no other matching M'' such that $M \preceq M''$ and $M'' \preceq M'$. The man-optimal and woman-optimal stable matchings appearing at the top and bottom of the Hasse diagram respectively.

The number of stable matchings that an SM instance admits can be exponentially large with respect to the size of the instance [56]. This implies that any algorithm that is designed to produce this set may need an exponentially large number of steps to complete. Thus enumerating the entire set of stable matchings has a worst case exponential complexity. However the average number of stable matchings grows at a much slower rate. Lennon and Pittel showed in [81] that the expected number of stable matchings in a random SM instance of size n taken uniformly out of the $(n!)^{2n}$ possibilities is of the order $n \ln n$.

Central to the construction of the set of all stable matchings, is the concept of *rotations*. A sequence of man-woman pairs in stable matching M can form a rotation ρ if, for the man in each pair in ρ, the woman in the next pair is the next woman on his list to

prefer him to her partner in M. In this arrangement, we consider the rotation as a cycle with the last pair being connected to the first. Intuitively this means that each man in the rotation can be unassigned to his partner in M and assigned to the woman in the next pair. Mathematically, a rotation $\rho = (m_{i_0}, w_{i_0}), (m_{i_1}, w_{i_1}), ..., (m_{i_r-1}, w_{i_r-1})$, *exposed* in a stable matching M, is an ordered list of pairs in M such that for each j $(0 \leq j \leq r-1)$, $m_{i_{j+1}} = next(m_{i_j})$ where $j+1$ is taken modulo r. We define $next(m_{i_j})$ as the man currently matched to the woman $s_{m_{i_j}} (= w_{i_{j+1}})$ who is the next woman on m_{i_j}'s list that prefers m_{i_j} to her partner in M. A rotation can be exposed in more than one stable matching. Considering an SM instance I, a rotation $\rho = (m_{i_0}, w_{i_0}), (m_{i_1}, w_{i_1}), ..., (m_{i_r-1}, w_{i_r-1})$ is said to be *eliminated* from a stable matching M when all men m_{i_j} who are in ρ are unassigned from their partners in M and matched with $w_{i_{j+1}}$ in ρ (again $j+1$ taken modulo r) with all other men retaining their partners in M. The new matching obtained, denoted by M/ρ, is also a stable matching in I.

The set of rotations that can be exposed in all stable matchings present in an SM instance forms a partial order based on a *precedence* relation. A rotation ρ *precedes* another ρ' (denoted by $\rho \lhd \rho'$) if ρ' cannot be exposed in a stable matching until ρ has been exposed and eliminated. Starting from the man-optimal matching M_0, rotations can be successively exposed and eliminated, yielding new stable matchings at each stage, until the woman-optimal matching M_z is reached. This gives all the rotations in the problem instance. Obviously an SM instance with a unique stable matching $(M_0 = M_z)$ will contain no rotations. It is also shown in [38] that if a rotation ρ is exposed in some stable matching M, then $M \preceq M/\rho$. This shows the relationship between the precedence relation between rotations and the dominance relationship between stable matchings [38]. Moreover, there is a one-one relationship between the subsets of the *rotation poset* that are closed under this precedence relation and the set of all stable matchings [56, 38]. Thus algorithms can be developed that exploit this correspondence (and structures related to the rotation poset) to construct the set of all stable matchings [36, 38].

The *rotation poset* is a pair $(R, \lhd)$ where R is the set of all rotations and $\lhd$ is the precedence relation on rotations. It can be constructed in $O(n^2)$ time using only the preference lists of the men and women [36]. The *rotation digraph*, a directed, acyclic graph whose edges correspond to a subset of the pairs of the rotation poset, can be used to design a range of efficient algorithms to find the set of all stable matchings and other useful structures. This idea that structures inherent to the matching problems can be constructed in polynomial time and used by efficient algorithms to produce various types of optimal matchings is very important. It motivates a considerable part of the development work into visualisation of these structures to be presented later in Chapter 8.

2.2.3 Stable Marriage problem with Incomplete lists

Another variant of SM that is of practical significance occurs when each agent is allowed to rank only a subset of the members of the opposite sex in his/her preference list. This typically happens in matching schemes where it is impractical or expensive to require agents to have complete preference lists. This problem is called the *Stable Marriage problem with Incomplete lists* (SMI) [38, Section 1.4.2]. Here an *acceptable pair* (m_i, w_j) is one in which w_j appears in m_i's preference list and m_i appears in w_j's preference list also. For SMI instances, the number of men and women in the instance need not be the same. A *matching* M is a set of acceptable pairs such that each agent appears in at most one pair. A matching does not necessarily contain all agents. A pair (m_i, w_j) forms a *blocking pair* with respect to M if (i) (m_i, w_j) is an acceptable pair, (ii) m_i is unmatched in M or prefers w_j to $M(m_i)$ and (iii) w_j is unmatched in M or prefers m_i to $M(w_j)$. A *stable* matching in this context is one that admits no blocking pair. The same set of agents are assigned in all stable matchings in a given instance of SMI [30]. This means that, although they need not be complete, all stable matchings given an instance of SMI have the same size. Moreover, every instance of SMI admits a stable matching and `Alg-EGS` can be easily extended to solve SMI instances in polynomial time [38, Section 1.4.2].

2.2.4 Stable Marriage problem with Ties

Another variant of SM that has significance in practical applications occurs when the agents' preference lists are allowed to contain *ties*. This variant of SM can arise in scenarios where the preferences are derived from, for example, performance scores where ties are a possibility. An example of this was the matching of medical students to hospital posts in England in 2005-2006 where applicants were ranked partly based on their academic results. On an agent's preference list, if two or more entries belong to a tie, the agent is indifferent between them. Each preference list can be regarded as a set of tied batches of agents such that (i) the agent is indifferent between all members of a tie and (ii) the agent prefers each member of a given tie to each member of any successor tie. We denote this problem as the *Stable Marriage problem with Ties* (SMT) [51]. In this context, the notion of stability needs to be redefined. Given an SMT instance I, a matching M in I can be evaluated in terms of *weak stability*, *strong stability* and *super stability* [51]. These three forms of stability are distinguished by the definitions of the blocking pairs they aim to prevent. Considering weak stability, a pair (m_i, w_j) blocks M if both m_i and w_j *strictly* prefer each other to their partners in M. Considering strong stability (m_i, w_j) blocks M if either (i) m_i strictly prefers w_j to his partner in M while w_j either strictly prefers m_i to her partner in M or is indifferent

between the two, or (ii) w_j strictly prefers m_i to her partner in M while m_i either strictly prefers w_j to his partner in M or is indifferent between the two. Considering super-stability, (m_i, w_j) blocks M if each of m_i and w_j either prefers the other to their partner in M or is indifferent between them. Figure 2.2 shows a sample SMT instance I (where brackets indicate ties).

men's preferences	women's preferences
$m_1 : (w_2 \quad w_4 \quad w_1 \quad w_3)$	$w_1 : m_2 \quad m_1 \quad m_4 \quad m_3$
$m_2 : w_3 \quad w_1 \quad w_4 \quad w_2$	$w_2 : (m_4 \quad m_3) \quad m_1 \quad m_2$
$m_3 : w_2 \quad (w_3 \quad w_1) \quad w_4$	$w_3 : m_1 \quad m_4 \quad m_3 \quad m_2$
$m_4 : (w_4 \quad w_1 \quad w_3) \quad w_2$	$w_4 : m_2 \quad (m_1 \quad m_4 \quad m_3)$

Figure 2.2: An instance of the Stable Marriage problem with Ties

Every instance of SMT admits a weakly stable matching which can be found by arbitrarily breaking the ties, thus reducing the problem to an SM instance which can then be solved using the Gale-Shapley algorithm [38]. The constructed stable matching is then weakly stable in the original SMT instance. However, SMT instances need not admit strongly stable or super-stable matchings. In the case of *complete indifference* where there are no strict preferences, a super-stable matching cannot exist. Figure 2.3 also shows an SMT instance that admits no strongly stable matching as pair (m_2, w_1) would block the matching $\{(m_1, w_1), (m_2, w_2)\}$ and pair (m_2, w_2) would block the matching $\{(m_i, w_2), (m_2, w_1)\}$. Algorithms for finding a strongly and super stable matching or reporting that none exists are described in [51]. These algorithms run in $O(n^4)$ and $O(n^2)$ respectively where n is the number of men or women in the instance.

men's preferences	women's preferences
$m_1 : w_1 \quad w_2$	$w_1 : m_2 \quad m_1$
$m_2 : (w_1 \quad w_2)$	$w_2 : m_2 \quad m_1$

Figure 2.3: An instance of the Stable Marriage problem with Ties

2.2.5 Stable Marriage problem with Ties and Incomplete lists

A generalisation of each of SMT and SMI arises when preference lists can simultaneously include ties and be incomplete, giving rise to the *Stable Marriage problem with Ties and Incomplete lists* (SMTI). The notions of weak, strong and super stability defined for SMT instances need to be refined slightly when applied to SMTI. Let (m_i, w_j) be an acceptable pair in an instance I of SMTI and let M be a matching in I.

1. In the case of weak stability, the pair (m_i, w_j) blocks M if
   (a) m_i is unmatched in M or prefers w_j to his partner in M, and
   (b) w_j is unmatched in M or prefers m_i to her partner in M.
2. In the case of strong stability, the pair (m_i, w_j) blocks M if either (i)
   (a) m_i is unmatched in M or prefers w_j to his partner in M, and
   (b) w_j is unmatched in M or prefers m_i to her partner in M or is indifferent between them.

   or (ii)
   (a) m_i is unmatched in M or prefers w_j to his partner in M or is indifferent between them, and
   (b) w_j is unmatched in M or prefers m_i to her partner in M.
3. In the case of super stability, the pair (m_i, w_j) blocks M if
   (a) m_i is unmatched in M or prefers w_j to his partner in M or is indifferent between them, and
   (b) w_j is unmatched in M or prefers m_i to her partner in M or is indifferent between them.

As in the case of SMT, SMTI instances need not admit strongly and super-stable matchings but every SMTI instance does admit a weakly stable matching which can again be found by breaking all ties arbitrarily and solving the resulting SMI instance. The sizes of weakly stable matchings in SMTI can vary depending on the manner in which the ties are broken [86]. This motivates the problems of finding a maximum and minimum weakly stable matching given an instance of SMTI (namely MAX SMTI and MIN SMTI respectively). Each problem is shown to be NP-hard by Manlove et al. [86] even in restrictive cases where ties appear on one set of preference lists only, the ties are at the tails of lists, there is at most one tie per list, and each tie is of length 2. Over the years research in SMTI and its many-to-one generalisation HRT has been very active. We describe these results in the context of HRT in Section 2.3.2.

2.2.6 Other optimality criteria

It is sometimes useful in applications to specify additional optimality criteria relative to stable matchings. These include:

1. *Egalitarian stable matchings* seek to optimise the satisfaction of both men and women simultaneously. If pair (m_i, w_j) is 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 list and the rank of w_j in M as the position of m_i on w_j's list. The *weight* of M is the sum of the ranks of all the men and women in M. An egalitarian stable matching has minimum weight over all the possible stable matchings. An efficient algorithm to find an egalitarian stable matching given an SM instance, which relies heavily on the distributive lattice structure of the set of all stable matchings, is described in [57].

2. *Minimum regret stable matchings* are the stable matchings in which the rank of the worst off person is minimised. An efficient algorithm for finding a minimum regret stable matching, given an instance of SM, is described in [36]

3. *Sex equal stable matchings* are stable matchings in which the absolute value of the difference between the sum of the ranks of all the men and the sum of the ranks of all the women is minimised. The problem of finding a sex-equal stable matching given an SM instance is NP-hard [71]. This was shown to be true even if the preference lists are of length at most 3 [91].

As stated earlier, although these optimality criteria can be obtained by enumerating the set of all stable matchings, more efficient algorithms to find these optimal stable matchings have been developed where possible [57, 36].

2.3 The Hospitals/Residents problem (HR)

2.3.1 Introduction

The *Hospital/Residents problem* (HR) [29] is a generalisation of SMI in which agents of one set (the hospitals) may be involved in one or more assignments in a given matching. A common application is the matching of residents (i.e., graduating medical students) to hospitals where each resident ranks a set of hospitals and each hospital ranks a set of residents in order of preference.

An instance I of HR involves a set $R = \{r_1, r_2, ..., r_{n_1}\}$ of *residents* and $H = \{h_1, h_2, ..., h_{n_2}\}$ of *hospitals*. Each resident $r_i \in R$ ranks a subset of H in strict order of preference with each hospital $h_j \in H$ ranking a subset of R, consisting of those residents who ranked h_j, in strict order of preference. Each hospital h_j also has a capacity $c_j \in \mathbb{Z}^+$ indicating the maximum number of residents that can be assigned to it. As in the case of SMI, a pair (r_i, h_j) is called an *acceptable pair* if h_j appears in r_i's preference list and vice versa. A *matching* M is a set of assignments, which are acceptable pairs,

such that each resident is assigned to at most one hospital and the number of residents assigned to each hospital does not exceed its capacity. We denote the hospital assigned to resident r_i in M as $M(r_i)$ (if r_i is matched in M) and the set of residents assigned to hospital h_j in M as $M(h_j)$. A resident r_i is *unmatched* in M if no acceptable pair in M contains r_i. A hospital h_j is *undersubscribed* in M if $|M(h_j)| < c_j$. Otherwise h_j is said to be *full* in M. An acceptable pair (r_i, h_j) *blocks* a matching M, or forms a *blocking pair* with respect to M, if r_i is either unmatched or prefers h_j to $M(r_i)$ and h_j is either undersubscribed or prefers r_i to at least one member of the set $M(h_j)$. A matching M is said to be *stable* if there is no blocking pair with respect to M.

This problem was also introduced by Gale and Shapley in [29] although in the context of college admissions. Every instance of HR admits at least one stable matching which can be found in linear time [29]. As described in the context of SM, matchings in the HR context can be hospital-optimal or resident-optimal depending on the orientation of the algorithm used. Given an instance I of HR, a stable matching M is said to be resident-optimal if every resident has their best possible partner in M taken over the set of all stable matchings in I. An analogous definition for hospital-optimal stable matchings exists. An HR instance may admit more than one stable matching, however, all stable matchings in a given HR instance have the same cardinality and contain the same set of residents [30, 101]. Figure 2.4 shows a sample HR instance I consisting of 6 residents and 3 hospitals with each hospital having a capacity of 2. The instance admits a stable matching $M = \{(r_1, h_2), (r_2, h_1), (r_3, h_1), (r_4, h_3), (r_6, h_2)\}$.

residents' preferences	hospitals' preferences
$r_1 : h_2 \quad h_1$	$h_1 : (2) : r_1 \quad r_3 \quad r_2 \quad r_5 \quad r_6$
$r_2 : h_1 \quad h_2$	$h_2 : (2) : r_2 \quad r_6 \quad r_1 \quad r_4 \quad r_5$
$r_3 : h_1 \quad h_3$	$h_3 : (2) : r_4 \quad r_3$
$r_4 : h_2 \quad h_3$	
$r_5 : h_2 \quad h_1$	
$r_6 : h_1 \quad h_2$	

Figure 2.4: An instance of the Hospitals/Residents problem

2.3.2 Hospital/Residents with Ties

A generalisation of HR occurs when the preference lists of the residents and hospitals are allowed to contain *ties*, thus forming the *Hospital/Residents Problem with Ties* (HRT). A tie in a resident's or hospital's preference list is defined analogously to the definition given for ties in the SMT context. A blocking pair can be defined with respect

to the different stability criteria outlined for SMT and SMTI above and matchings can again be considered in terms of weak stability, strong stability and super stability. In the case of weak stability both the resident and hospital involved in a blocking pair must be better-off before a matching can be undermined. In the case of strong stability either the resident or hospital must be better off with none of them being worse-off. In the case of super stability, a resident-hospital pair can block a matching as long as both are not worse-off. Formal definitions of these stability criteria in the HRT context are as follows. Let (r_i, h_j) be an acceptable pair in an instance I of HRT and let M be a matching in I.

1. In the case of weak stability, the pair (r_i, h_j) blocks M if
   (a) r_i is unmatched in M or prefers h_j to her assigned hospital in M, and
   (b) h_j is undersubscribed in M or prefers r_i to its worst assigned resident in M.
2. In the case of strong stability, the pair (r_i, h_j) blocks M if either (i)
   (a) r_i is unmatched in M or prefers h_j to her assigned hospital in M, and
   (b) h_j is undersubscribed in M or prefers r_i to its worst assigned resident in M or is indifferent between them.

   or (ii)

   (a) r_i is unmatched in M or prefers h_j to her assigned hospital in M or is indifferent between them, and
   (b) h_j is undersubscribed in M or prefers r_i to its worst assigned resident in M
3. In the case of super stability, the pair (r_i, h_j) blocks M if
   (a) r_i is unmatched in M or prefers h_j to her assigned hospital in M or is indifferent between them, and
   (b) h_j is undersubscribed in M or prefers r_i to its worst assigned resident in M or is indifferent between them.

Every instance of the HRT problem admits at least one weakly stable matching. This can be obtained by breaking the ties in both sets of preference lists in an arbitrary manner, thus giving rise to a HR instance which can then be solved using variants of the Gale-Shapley algorithm for HR [38]. However, in general, the way in which the ties are broken yields stable matchings of varying sizes [86] and the problem of finding a maximum weakly stable matching given an HRT instance (MAX HRT) is known to be NP-hard [86] as HRT is a many-to-one generalisation of SMTI.

Research in HRT and SMTI and their variants has been very active since the initial results were published in [84, 64]. In particular, weak stability has attracted considerable attention perhaps due to the fact that under weak stability, HRT and SMTI instances are guaranteed to admit a stable matching though this need not be the case for strong stability and super stability [51]. Various approximation algorithms for MAX HRT can be found in the literature [66, 67, 74, 86, 59] with the best having a bound of 3/2 [90, 75, 97]. The parameterized complexity of MAX HRT has also been studied in the literature. In [88] MAX HRT was shown to be in FPT with the parameter being the sum of the lengths of the ties in the preference lists.

Concerning inapproximability results, Halldorsson et al. showed that it is NP-hard to approximate MAX SMTI to within δ for some $\delta > 1$ [39] (where δ is very close to 1) even if each man's preference list is of length at most 7 and each woman's preference list is of length at most 4 and each preference list is derived from two master lists of men and women. Irving et al. presented the same inapproximability result for the restriction where men's and women's preference lists are of length at most 3 and 4 respectively [61]. In the case of unbounded length preference lists where each man's preference list is strictly ordered and each woman's preference list is either strictly ordered or is a tie of length 2, MAX SMTI was shown to be inapproximable to within $21/19 - \varepsilon$ for some $\varepsilon > 0$ unless P=NP [40]. Finally Yanagisawa improved this lower bound to 33/29 for the case where ties are on both sides and each tie is of length 2 [115]. Various heuristics for solving MAX HRT have also been developed [60, 34, 35].

2.3.3 Cloning HR instances

A technique known as *cloning* can be used to convert an HR instance I into an instance an SMI I' in polynomial time, such that there is a bijective function between the sets of stable matchings in I and I' [38, 106]. Let I be an instance of HR where $R = \{r_1, r_2, ..., r_{n_1}\}$ is the set of residents and $H = \{h_1, h_2, ..., h_{n_2}\}$ is the set of hospitals. Let c_j be the capacity of hospital $h_j \in H$. We can construct an instance I' of SMI as follows. Each resident in I corresponds to a man in I'. Each hospital $h_j \in H$ gives rise to c_j women in I', denoted by $h_{j,1}, h_{j,2}, ..., h_{j,c_j}$, each of whom has the same preference list as h_j in I' but with a capacity of 1. Each man $r_i \in R$ starts off with the same preference list in I' as he has in I. We then replace each hospital h_j on his list by the c_j women $h_{j,1}, h_{j,2}, ..., h_{j,c_j}$ listed in strict order (increasing on second subscript). There is a bijective function between the sets of stable matchings in I and I'. This relation between HR and SMI was shown to hold (but not necessarily with a bijective function) when cloning MAX HRT to MAX SMTI instances in [85] thus some of the positive results for MAX SMTI can carry over to MAX HRT. The same technique will be used in Section

4.5.1 for cloning another variant of HR to its one-to-one special case.

2.4 The Stable Roommates problem (SR)

The *Stable Roommates problem* (SR) [29] involves an even-sized set $R = \{a_1, a_1, ..., a_n\}$ of agents with each agent $a_i \in R$ ranking the other agents $R \backslash \{a_i\}$ in strict order of preference. An agent a_i *prefers* some a_j to another a_k if a_j appears before a_k in a_i's preference list. While SM and HR instances can be modelled as bipartite graphs involving two disjoint sets of agents, SR instances the corresponding graph is not bipartite. A *matching* M in this context is a set of $n/2$ unordered pairs of agents where each agent appears in at most one pair. We denote the agent paired with a_i in M as $M(a_i)$. A pair of agents $\{a_i, a_j\}$ *blocks* M, or forms a *blocking pair* with respect to M, if a_i prefers a_j to $M(a_i)$ and a_j prefers a_i to $M(a_j)$. A matching is *stable* is it admits no blocking pair. Figure 2.5 shows a sample SR instance I. It has been shown that an instance of SR need not admit a stable matching [29]. Indeed this is true of the instance in Figure 2.5. An $O(n^2)$ algorithm also exists in the literature to find a stable matching given an instance of SR if one exists [50].

$$
\begin{array}{llll}
a_1 : a_3 & a_2 & a_4 \\
a_2 : a_1 & a_3 & a_4 \\
a_3 : a_2 & a_1 & a_4 \\
a_4 : a_3 & a_2 & a_1
\end{array}
$$

Figure 2.5: An instance of the Stable Roommates problem

The general idea behind the algorithm presented in [50] is to successively remove entries from the preference lists until either some agent's preference list is empty, in which case no stable matching exists (i.e., the instance is *unsolvable*), or all the preference lists are left with a single entry, in which case the resulting assignment will be a stable matching. The algorithm runs in two phases. The first phase removes entries from the preference list that cannot be involved in any stable matching according to a proposal sequence that is similar to the Gale-Shapley algorithm. If after this phase, some agent is left with an empty preference list then the instance is unsolvable. If all the preference lists after the first phase contain only a single entry, then the resulting matching, obtained by simply assigning every agent to the single entry on their preference list, is the unique stable matching. If however some preference lists contain more than one entry further work needs to be done in order to obtain a stable matching. The second phase continues to trim the preference lists until either all preference lists contain only one

entry or some agent's list becomes empty in which case the instance in unsolvable. This algorithm relies on the exposure and elimination of rotations as they are discovered. Rotations in the SR context are discussed in the next subsection.

2.4.1 Rotations in the Stable Roommates problem

The concept of a rotation in SR is similar to that described in the SM context. A notable difference between rotations in both cases is that for SM, a rotation is defined relative to a stable matching in which it is exposed while for SR, a rotation is defined relative to the *preference table* in which it is exposed. A preference table in an SR instance is a set of preference lists from which zero or more pairs have been deleted. These deletions take place during the first or second phases of the algorithm described in [50]. In the resulting preference table, if a_i is the first agent on a_j's preference list, then a_j is the last on a_i's. If for some preference table T, $f_T(a_i)$ and $s_T(a_i)$ are the first and second entries of agent a_i in T, then a *rotation* ρ, which is a set of pairs $(a_{i_0}, a_{j_0}), (a_{i_1}, a_{j_1}), ..., (a_{i_{r-1}}, a_{j_{r-1}})$ such that $a_{j_k} = f_T(a_{i_k})$ and $a_{j_{k+1}} = s_T(a_{i_k})$ for all k, $0 \leq k \leq r-1$, where $k+1$ is taken modulo r, is *exposed* in T. A rotation $\rho = (a_{i_0}, a_{j_0}), (a_{i_1}, a_{j_1}), ..., (a_{i_{r-1}}, a_{j_{r-1}})$ is said to be *non-singular* if $\bar{\rho} = (a_{j_1}, a_{i_0}), (a_{j_2}, a_{i_1}), ..., (a_{j_{r-1}}, a_{i_{r-2}}), (a_{j_0}, a_{i_{r-1}})$ is also a rotation. Rotation ρ is called the *dual* of $\bar{\rho}$ and vice versa. If $\bar{\rho}$ is not a rotation then ρ is said to be *singular*. Singular rotations need to be exposed and eliminated before any stable matching is found while non-singular rotations form a partial order which can be expoloted to find the set of all stable matchings.

2.4.2 Structure of stable matchings in SR

In SM, there is a clear structure contained within any instance: the set of all stable matchings in an SM instance forms a distributive lattice under a dominance relation [76]. In the SR context, the stable matchings form a semi-lattice (a weaker structure than the lattice structure) under a dominance relation [37]. The semi-lattice has just a single operation, meet or join, and is closed under this operation. It generalises the lattice structure in the SM case.

2.4.3 Stable Roommates with Incomplete lists

A more general version of SR allows for the preference lists to be incomplete, i.e., each agent ranks a subset of $R \backslash \{a_i\}$ in order of preference. We refer to this problem as the *Stable Roommates problem with Incomplete lists* (SRI). In this form, the number of agents need not be even. A pair of agents $\{a_i, a_j\}$ is an *acceptable pair* if a_i appears

in a_j's preference list and vice versa. A matching M is a set of acceptable pairs where each agent appears in at most one pair. The definition of a blocking pair also changes slightly. An acceptable pair $\{a_i, a_j\}$ forms a blocking pair with respect to M if a_i is unmatched or prefers a_j to $M(a_i)$, and a_j is unmatched or prefers a_i to $M(a_j)$. The algorithm for finding a stable matching given an SR instance (should one exist) can be extended to the SRI case [38, Section 4.5.2]. Figure 2.6 shows a sample SRI instance I with stable matching $M = \{\{a_1, a_2\}, \{a_3, a_4\}\}$.

$$
\begin{array}{llll}
a_1 : & a_2 & a_4 & a_3 \\
a_2 : & a_1 & a_3 & \\
a_3 : & a_2 & a_1 & a_4 \\
a_4 : & a_3 & a_1 &
\end{array}
$$

Figure 2.6: An instance of the Stable Roommates problem with Incomplete Lists

2.5 Locally Stable Matchings

2.5.1 Introduction

Although the concept of stability is important in many applications of matching problems, there are classes of matching problems (such as the Stable Roommates problem) for which an instance is not guaranteed to admit a stable matching [29]. Moreover, enforcing the stability requirement tends to reduce the size of the matchings discovered [15]. This is an issue particularly in the case of applications where it is desired to find a largest possible matching. It may be argued that in some applications where the size of the matching produced is just as important (if not more important) than the stability of the matching, relaxing the stability definition to allow for larger matchings is justified.

One approach taken was to weaken the stability criterion by tolerating a number of blocking pairs. The smaller the number of blocking pairs, the more stable a matching is said to be. The problem then becomes to find a maximum size matching with as few blocking pairs as possible. Such a model of "almost" stable matchings was considered in [15]. The authors identified a number of practical applications in which such an approach would be feasible. They defined the problem of finding a maximum matching with the minimum number of blocking pairs given an SMI instance as MAX SIZE MIN BP SMI. They showed that the problem is NP-hard to approximate to within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ where n is the number of men in a given instance. They also showed the

problem to be NP-hard to approximate to within δ for some $\delta > 1$ even if all preference lists are of length at most 3. They then provideed a polynomial-time algorithm for the special case in which the preference lists of one sex are of length at most 2. Hamada et al. strengthened this inapproximability bound to within $n^{1-\varepsilon}$ even if all preference lists are of length at most 3 [42].

Another approach is to leverage any information about the social relationship between agents in the matching market. Instead of trying to minimise the number of blocking pairs, we may redefine the stability criterion to allow for certain blocking pairs that are unlikely to lead to a matching being undermined. It is generally assumed that, in the HR context, a resident-hospital pair that blocks a matching in theory will also block the matching in practice. However this assumption is not always true in some real-world applications, as resident-hospital pairs are more likely to form blocking pairs in practice if social ties exist between them. This observation, coupled with the need to find the largest possible matchings, have motivated studies into alternative, weaker stability definitions that still aim to prevent a given matching from being subverted while increasing the number of agents involved in the matchings. If social ties do not exist between a pair of agents, it is assumed that they are unlikely to subvert a matching in practice as they may not become aware that they could form a blocking pair with one another in reality. They are therefore not considered as blocking pairs. Redefining stability on the bases of agents who are most likely to form blocking pairs in practice can yield larger matchings whilst still providing a degree of robustness against a matching being undermined. This section discusses this concept, which we call *social stability* (also referred to as *local stability* in the literature), in the context of SM and HR as well as similar ideas described in the literature.

2.5.2 Locally Stable Matchings in the Job Market Context

Arcaute and Vassilvitskii [11] studied the Hospitals/Residents problem in the context of assigning job applicants to company positions. They observed that applicants are more likely to be employed by a company if they are recommended by their friends who are already employees of that company. Given the large amount of applications that may be submitted for vacancies, companies are increasingly reliant on personal in-house recommendations for making employment decisions. In their model applicants have a limited knowledge of the matching market and can only form blocking pairs with companies in which their friends are employees. In the context of stable matching theory, an applicant-company pair (a, c) blocks a matching M if (a, c) blocks it in the traditional sense (as described in the analogous HR context) and a *is friends with* another applicant a' assigned to c in M. Thus their problem (which was later called HR+SN

by Cheng and McDermid [22]) incorporates both the traditional HR problem and additionally an underlying social network, represented as an undirected graph consisting of applicants as nodes and edges between nodes where the corresponding applicants have some social ties (e.g., are friends). They call matchings that admit no blocking pair in this context *locally stable* due to the addition of the informational constraint on blocking pairs. Concerning the static properties of locally stable matchings, Arcaute and Vassilvitskii [11] showed that the set of locally stable matchings does not form a distributive lattice. They also considered a dynamic version of their model (where applicants and/or companies arrive and are matched over time), providing a decentralised version of the Gale-Shapley algorithm for finding a locally stable matching.

Cheng and McDermid [22] investigated the problem further and presented some algorithmic results. They showed that locally stable matchings can be of different sizes and MAX HR+SN the problem of finding a maximum locally stable matching is NP-hard. They identified special cases where the problem is polynomially solvable and gave upper and lower bounds on the approximability of the problem. They showed that when the social network is a complete graph, local stability and classic stability become the same thus finding a maximum locally stable matching is polynomially solvable. If the social network is an empty graph, then every maximum cardinality matching is a locally stable matching thus finding a maximum locally stable matching is also computationally easy. Concerning the approximability of MAX HR+SN, they observed that every stable matching in the classical sense is also a locally stable matching and is at least $1/2$ the size of a maximum locally stable matching. They also proved that it is NP-hard to approximate MAX HR+SN to within $21/19 - \delta$ for any $\delta > 0$.

The problem of finding an approximation algorithm for MAX HR+SN with a performance guarantee better than 2 still remains open. Given the close relationship between MAX HR+SN and MAX HRT and the fact that MAX HRT can be approximated to within $3/2$, Cheng and McDermid conjectured that a $3/2$ approximation algorithm for MAX HR+SN also exists. They proposed a strategy similar to those used for MAX HRT in [75, 90]. They concluded the paper by describing a special case where social ties exist only between applicants who are matched in every stable matching and applicants who are not. In this case every stable matching is a $3/2$ approximation of a maximum locally stable matching. Askalidis et al. [13] suggested that the elusiveness of a $3/2$ approximation algorithm for MAX HR+SN is due to the fact that the model is dynamic in nature – blocking pairs are not determined from the problem instance alone but change depending on the matching being considered.

2.5.3 Other results relating to Locally Stable Matchings

Hoefer and Wagner [44, 45] studied a more general version of local stability. In their model, the social network graph involves all agents and need not be bipartite. A pair *locally blocks* a given matching M if (i) it blocks in the classical sense, and (ii) the agents involved are at most l edges apart in the social network graph augmented by M. This scenario can be viewed as a generalisation of HR+SN ($l = 2$) where blocking pairs can only be formed through a third party and other models where blocking pairs can be formed through direct communication between the agents involved (i.e. $l = 1$). They also established a lower bound for the approximabiliy of the problem of finding a maximum locally stable matching (for the case that $l \leq 2$).

This idea of local stability has been investigated in the context of the Stable Roommates problem (a non-bipartite generalisation of the Stable Marriage problem) in [19]. Here, the *Stable Roommates problem with Free edges* (SRF) as introduced was motivated by the observation that, in kidney exchange matching schemes, donors and recipients do not always have full information about others and are more likely to have information only on others in the same transplant centre as them. The problem is defined by the traditional Stable Roommates problem together with a set of free pairs (or edges in the underlying graph). These correspond to pairs of agents in different transplant centres that do not share preference information who may be involved in stable matchings, but cannot block any matching. It is shown in [19] that the problem of determining whether a locally stable matching exists, given an SRF instance, is NP-complete.

2.6 The Student/Project Allocation problem

2.6.1 Introduction

In most academic programmes students are usually required to take up individual or group projects offered by lecturers. Students may be required to rank a subset of the projects they find acceptable in order of preference. Each project is offered by a unique lecturer who may also be allowed to rank the projects she offers or the students who are interested in taking her projects in order of preference. Each student can be assigned to at most one project and there are usually constraints on the maximum number of students that can be assigned to each project and lecturer. The problem then is to assign students to projects in a manner that satisfies these capacity constraints while taking into account the preferences of the students and lecturers involved. This problem has been described in the literature as the *Student-Project Allocation problem* (SPA) [8, 87, 9, 68]. Variants of SPA also exist in which *lower quotas* are assigned to projects

and/or lecturers. These lower quotas indicate the minimum number of students to be assigned to each project and lecturer.

Although described in an academic context, applications of SPA need not be limited to assigning students to projects but may extend to other scenarios, such as the assignment of employees to posts in a company where available posts are offered by various departments. Applications of SPA in an academic context can be found at the University of Glasgow [116], the University of York [25, 72, 112], the University of Southampton [10, 43] and the Geneva School of Business Administration [113]. As previously stated, it is widely accepted that matching problems (like SPA) are best solved by centralised matching schemes where agents submit their preferences and a central authority computes an optimal matching that satisfies all the specified criteria [38]. Moreover the potentially large number of students and projects involved in these schemes motivates the need to discover efficient algorithms for finding optimal matchings.

In SPA, students are always required to provide preference lists containing projects. However, variants of the problem may be defined depending on the presence and nature of lecturer preference lists. Some variants of SPA, as discussed in Section 2.6.2, require both students and lecturers to provide strictly ordered preference lists. In these cases, the well-known stability criterion is applied on the matchings produced. In other variants of SPA, which we introduce in Section 2.6.3, only students are required to produce preference lists in which case other feasibility and optimality criteria need to be considered. In general, SPA models may vary from one application of the problem to another. In Section 2.6.4 we discuss some of these variants that appear in the literature.

2.6.2 Two-sided preferences and stability

In one variant of SPA, each lecturer is required to rank the students who find at least one of her offered projects acceptable, in strict order of preference, thus forming the *Student/Project Allocation problem with lecturer preferences over Students* (SPA-S) [8]. A lecturer's preference list will typically reflect her assessment of the ability of these students. Another variant of SPA involves lecturers ranking the projects they offer in strict order of preference thus leading to the *Student/Project Allocation problem with lecturer preferences over Projects* (SPA-P) [87, 68]. It may be expected that lecturers will typically give higher priority to projects that are closely related to their research interests. Another variant that has been of interest in the literature is a generalisation of each of the SPA-S and SPA-P problems. In this problem lecturers rank student-project pairs in strict order of preference, thus leading to the *Student/Project Allocation problem with lecturer preferences over Student-Project pairs* (SPA-(S,P)) [8, 9]. Each

student-project pair on a lecturer's preference list involves a project offered by the lecturer and a student who finds that project acceptable. This allows lecturers to not only estimate the suitability of students to all her offered projects but to specific projects as well. A formal definition of SPA-(S,P) as well as the conditions necessary for a student-project pair to block a matching were presented in [9].

These variants of SPA involving lecturer preference lists have been studied relative to several definitions of stability. In the SPA context, the stability objective is to produce a matching M in which no student-project pair (s_i, p_j) exists such that (s_i, p_j) are not matched in M, and s_i and the lecturer offering p_j can simultaneously improve by being paired together (i.e., by adding (s_i, p_j) to M). This process could potentially cause them to abandon their previous partners in M. As is the case with other matching problems involving preferences, such student-project pairs are called *blocking pairs*. The formal definition of stability may vary depending on the SPA variant being considered.

Abraham et al. [8] gave an $O(m)$ algorithm for finding a stable matching given an SPA-S instance where m is the number of student-project pairs in the instance. They observed that, given a SPA-S instance, such a matching is bound to exist. Their algorithm produces the *student-optimal* stable matching which is the stable matching in which each student obtains their best possible project over all stable matchings. They also presented an algorithm for finding the *lecturer-optimal* stable matching in which each lecturer obtains the best set of students she can obtain in any stable matching given a SPA-S instance.

In the SPA-P context a matching may be undermined not only by student-project blocking pairs but also groups of students who choose to cooperate for their collective benefit (called *blocking coalitions*). Given a matching M, a coalition is a set of students $\{s_{i_0}, ..., s_{i_{r-1}}\}$, for some $r \geq 2$, each of whom is assigned in M, such that s_{i_j} prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$ $(0 \leq j \leq r-1$, where addition is taken modulo $r)$. That is, the students in the coalition could permute the projects that they have been assigned to in M so as to be better off. Thus a matching is stable if it avoids both blocking pairs and blocking coalitions. Given a SPA-P instance, a stable matching is guaranteed to exist and can be found in $O(m)$ time [87] where m is the total length of the preference lists in the instance. However, given a SPA-P instance, stable matchings may be of varying sizes and the problem of finding a maximum stable matching was shown to be NP-hard [87] although approximable to within 3/2 but not to within $21/19 - \varepsilon$ unless P=NP [68].

Abu El-Atta and Moussa showed that an instance of SPA-(S,P) is guaranteed to admit a stable matching and that such a matching can be found in $O(m)$ time where m is the total length of the students' preference lists. A full description of the results relating

to these SPA variants with lecturer preferences can be found in [85].

2.6.3 One-sided preferences and profile-based optimality

In many practical SPA applications it is considered appropriate to allow only students to submit preferences over projects. When preferences are specified by only one set of agents in a two-sided matching problem, the notion of stability becomes irrelevant. This motivates the need to adopt alternative solution criteria when lecturer preferences are not allowed. In this subsection we describe some of these solution criteria and briefly present results relating to them. These criteria consider the size of the matchings produced as well as the satisfaction of the students involved.

When preference lists of lecturers are absent, the SPA problem becomes a two-sided matching problem with one-sided preferences. We assume students' preference lists can contain ties in these SPA variants. Various optimality criteria for such problems have been studied in the literature [85]. Some of these criteria depend on the *profile* or the *cost* of a matching. In the SPA context, the *profile* of a matching is a vector whose rth component indicates the number of students obtaining their rth-choice project in the matching. The *cost* of a matching (w.r.t. the students) is the sum of the ranks of the assigned projects in the students' preference lists (that is, the sum of rx_r taken over all components r of the profile, where x_r is the rth component value).

A *minimum cost maximum matching* is a maximum cardinality matching with minimum cost. A *rank-maximal matching* is a matching that has lexicographically maximum profile [55, 53]. That is the maximum number of students are assigned to their first-choice project and subject to this, the maximum number of students are assigned to their second choice project and so on. However a rank maximal matching need not be a maximum matching in the given instance (see, e.g., [85, p.43]). Since it is usually important to match as many students as possible, we may first optimise the size of the matching before considering student satisfaction. Thus we define a *greedy maximum matching* [54, 93, 48] to be a maximum matching that has lexicographically maximum profile. The intuition behind both rank-maximal and greedy maximum matchings is to maximize the number of students matched with higher ranked projects. This may lead to some students being matched to projects that are relatively low on their preference lists. An alternative approach is to find a *generous maximum matching* which is a maximum matching in which the minimum number of students are matched to their Rth-choice project (where R is the maximum length of any students' preference list) and subject to this, the minimum number of students are matched to their $(R-1)$th-choice project and so on. Greedy and generous maximum matchings have been used to assign students to projects in the School of Computing Science, and students to elective

students' preferences:	lecturers' offerings:
$s_1 : p_1 \quad p_2 \quad p_3$	$l_1 : \{p_1, p_2\}$
$s_2 : p_1$	$l_2 : \{p_3\}$
$s_3 : p_2 \quad p_3$	project capacities: $c_1 = 1, c_2 = 1, c_3 = 1$
	lecturer capacities: $d_1 = 2, d_2 = 1$

Figure 2.7: A SPA instance I

courses in the School of Medicine, both at the University of Glasgow, since 2007. Figure 6.1 shows a sample SPA instance with greedy and generous maximum matchings, namely $M_1 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\}$ and $M_2 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ respectively.

A special case of SPA, where each project is offered by a unique lecturer with an infinite upper quota and zero lower quota, can be modelled as the *Capacitated House Allocation problem with Ties* (CHAT). This is a variant of the well-studied *House Allocation problem* (HA) [49, 117] which involves the allocation of a set of indivisible goods (which we call houses) to a set of applicants. In CHAT, each applicant is required to rank a subset of the houses in order of preference with the houses having no preference over applicants. The applicants play the role of students and the houses play the role of projects and lecturers. As in the case of SPA, we seek to find a many-to-one matching comprising applicant-house pairs. Efficient algorithms for finding profile-based optimal matchings in CHAT have been studied in the literature [48, 54, 110, 93]. The most efficient of these is the $O(R^* m\sqrt{n})$ algorithm for finding rank-maximal, greedy maximum and generous maximum matchings in CHAT problems due to Huang et al. [48] where R^* is the maximum rank of any applicant in the matching, m is the sum of all the preference list lengths and n is the total number of applicants and houses. These models however fail to address the issue of load balancing among lecturers. In order to keep the assignment of students fair each lecturer will typically have a minimum (lower quota) and maximum (capacity/upper quota) number of students they are expected to supervise. These numbers may vary for different lecturers according to other administrative and academic commitments.

The CHAT algorithms mentioned above are based on modelling the problem in terms of a bipartite graph with the aim of finding a matching in the graph which satisfies the stated criteria. However a more flexible approach would be to model the problem as a network with the aim of finding a flow that can be converted to a matching which satisfies the stated criteria. SPA has also been investigated in the network flow context [5, 116] where a *minimum cost maximum flow algorithm* is used to find a minimum cost

maximum matching and other profile-based optimal matchings. The model presented in [116] allows for lower quotas on lecturers and projects as well as alternative lecturers to supervise each project. By an appropriate assignment of edge weights in the network it is shown that a minimum cost maximum flow algorithm (due to Orlin [96]) can find rank maximal, generous maximum and greedy maximum matchings in a SPA instance. This takes $O(m \log n(m + n \log n))$ time in the worst case, where m and n are the number of vertices and edges in the network respectively. In the SPA context this takes $O(m_2^2 \log n_1 + m_2 n_1 \log^2 n_1)$ time where n_1 is the numbers of students and m_2 is the sum of all the students' preference list lengths. However this approach involves assigning exponentially large edge weights (see, e.g., [85, p.405]), which may be computationally infeasible for larger problem instances due to floating point inaccuracies in dealing with such high numbers. For example given a large SPA instance involving say, $n_1 = 100$ students each ranking $R = 10$ projects in order of preference, edge weights could potentially be of the order $n_1^R = 100^{10} = 10^{20}$ (and arithmetic involving such weights could easily require more than the 15-17 significant figures available in a 64-bit double-precision floating representation). Since the flow algorithms involve comparing these edge weights, floating point precision errors could easily cause them to fail in practice. Moreover using the standard assumption that arithmetic on numbers of magnitude $O(n_1)$ takes constant time, arithmetic on edge weights of magnitude $O(n_1^R)$ would add an additional factor of $O(R)$ onto the running time of Orlin's algorithm (as R memory locations will be needed to store each of these exponential edge weights in practice).

In Chapter 6 we present efficient algorithms for finding optimal matchings to SPA problems based on the profile-based greedy maximum and generous maximum optimality criteria. Our model allows for lecturer upper and lower quotas and finds these profile-based optimal matchings without the need for exponentially-large edge weights. Our algorithms run in $O(n_1^2 R m_2)$ time. Formal definitions of SPA and its variants are also presented in Chapters 6 and 7.

2.6.4 Other SPA models and approaches

The variants of SPA already discussed above have been motivated by both practical and theoretical interests. These variants are usually distinguished by the (i) feasibility and (ii) optimality criteria specific to them. In this section, we discuss some more SPA models found in the literature as well as other approaches that have been used to solve these problems. The techniques employed include *Integer Programming* (IP)[1] [10, 113, 109, 70, 109, 70], *Constraint Programming* (CP) [25, 112], and others including goal programming and genetic algorithms [111, 43, 80].

[1] which we will discuss in Section 2.7.

In [109], an IP model for SPA was presented with the aim of optimising the overall satisfaction of the students and the lecturers offering the projects (i.e., minimising the overall cost on both sides). In [10] an IP model was presented for SPA problems involving individual and group projects. Various objective functions were also employed (often in a hierarchical manner). These include minimising the cost, balancing the work-load among lecturers, maximising the number of students assigned and maximising the number of first-choice assignments (w.r.t. student preferences). In [113] a more general IP model for SPA which allows project lower quotas was also presented. However none of these models simultaneously considers profile-based optimality as well as upper and lower quota constraints.

In all the variants of SPA discussed above, very little work has been done in implementing and evaluating their performance empirically. In cases where multiple algorithms/approaches exist to solve the same problem, there does not seem to have been a previous systematic empirical evaluation comparing the performance of these approaches. To answer these questions, multiple random instances of the problems need to be generated and solved and the results analysed. We present results from empirical evaluations of some of these algorithms in Chapters 6 and 7.

2.7 Integer Programming approaches to matching problems

Linear Programming (LP) formulations for SM and its variants have been widely studied in the literature [114, 38, 108, 102]. These approaches usually involve constructing a set of linear inequalities J from an instance I of the matching problem such that there is a 1–1 correspondence between the set of stable matchings in I and the extreme points of the polytope specified by J. Vande Vate's approach involved constructing these linear inequalities (which describe the stable matchings) using the preference lists of the agents in I [114]. He also showed that, for SM, the extreme points of the solution polytopes of his LP are all integer-valued. This has also been shown to be true in the SMI and HR cases [108, 102] as well as in the SRI case [4]. However this result does not extend naturally to other stable matching problems, thus the need to impose further restrictions on the domains of the variables involved in J. In particular we may require that all variables have integer domains thus forming *Integer Programming* (IP) models.

Although finding an integer solution to a set of linear inequalities is known to be NP-hard [69, 31], current commercial and open-source IP solvers allow for reasonably-sized IP instances to be solved quickly. This has led to the development of various IP models for matching problems [109, 10, 105, 6, 113, 70, 20, 89, 17]. The problems studied

in these context have typically been shown to be NP-hard thus further justifying the use of IP techniques. An IP formulation for MAX SMTI was presented by Podhradskỳ [100] where he compared the sizes of weakly stable matchings obtained by running various approximation algorithms on SMTI instances against the maximum weakly stable matchings obtained from his IP formulation. Since his model had no proofs or detailed explanations, it is difficult to comment on its correctness. In Chapter 3 we present a compact IP model for the more general MAX HRT problem and prove its correctness. We also use IP as a way of verifying the correctness of implementations of various efficient algorithms throughout this thesis.

Chapter 3

An Integer Programming Approach to the Hospitals/Residents Problem with Ties

3.1 Introduction

In Chapter 2 we discussed SMTI and HRT, which are two-sided matching problems where each agent's preference list may contain ties. We gave a review of various stability criteria that can be applied to these problems: namely weak, strong and super-stability. In practice weak stability is the most commonly-studied stability concept due to the guaranteed existence of weakly stable matchings; by contrast strongly stable and super-stable matchings need not exist in a given problem instance. For a given instance of SMTI, weakly stable matchings may be of different sizes and furthermore, MAX SMTI, the problem of finding a maximum weakly stable matching given an SMTI instance has been shown to be NP-hard [86]. A variety of techniques ranging from approximation algorithms and heuristics, to constraint programming and integer programming [99] have been developed to cope with this hardness in practice. In this chapter we present a new integer programming formulation for MAX HRT and discuss various techniques that can be used to improve the performance of the model in practice.

This chapter is structured as follows. Section 3.2 presents our IP model for MAX HRT along with a proof of its correctness. In Section 3.3 we provide some details on the implementation of the model. Here we give algorithms that can help reduce the size of the IP model without affecting its accuracy. We also discuss other performance-improving techniques that can be applied to the model. Section 3.4 summarises some of the results obtained by evaluating the model against real-world and randomly generated

problem instances. Finally in Section 3.5 we conclude by highlighting some interesting open problems.

3.2 An IP model for MAX HRT

In this section we describe an IP model for MAX HRT. Let I be an instance of HRT consisting of a set $R = \{r_1, r_2, ..., r_{n_1}\}$ of residents and $H = \{h_1, h_2, ..., h_{n_2}\}$ of hospitals with each hospital h_j having capacity of c_j. We use the binary variable $x_{i,j}$ $(1 \leq i \leq n_1, 1 \leq j \leq n_2)$ to represent an acceptable pair in I formed by resident r_i and hospital h_j. Variable $x_{i,j}$ will indicate whether r_i is matched to h_j in a solution or not: if $x_{i,j} = 1$ in a given solution $\mathbf{x}$ then r_i is matched to h_j, otherwise r_i is not matched to h_j. We define $rank(r_i, h_j)$, the rank of h_j on r_i's preference list to be $k + 1$ where k is the number of hospitals that r_i strictly prefers to h_j. An analogous definition for $rank(h_j, r_i)$ holds. Obviously for HRT instances, agents in the same tie have the same rank. We define $rank(r_i, h_j) = rank(h_j, r_i) = \infty$ for an unacceptable pair (r_i, h_j). With respect to a pair (r_i, h_j), we define the sets:

$$T_{i,j} = \{r_p \in R : rank(h_j, r_p) \leq rank(h_j, r_i)\}.$$

$$S_{i,j} = \{h_q \in H : rank(r_i, h_q) \leq rank(r_i, h_j)\}.$$

Intuitively $T_{i,j}$ is the set of residents that h_j either prefers to or is indifferent with r_i. If h_j is matched to c_j residents in $T_{i,j}$ in a matching M then (r_i, h_j) cannot block M. Also $S_{i,j}$ is the set of hospitals that r_i either prefers to or is indifferent with h_j. If r_i is matched to any hospital in $S_{i,j}$ in M then (r_i, h_j) cannot block M. We also define the set $P(r_i)$ to be the set of hospitals that r_i finds acceptable and $P(h_j)$ to be the set of residents that h_j finds acceptable.

Figure 3.1 shows the resulting model. Constraint 1 ensures that each resident is matched to at most one hospital and Constraint 2 ensures that each hospital does not exceed its capacity. Finally Constraint 3 ensures that the matching is stable by ruling out the existence of any blocking pair. In order to ensure that an acceptable pair (r_i, h_j) does not form a blocking pair, Constraint 3 enforces the following rules:

1. if h_j is not matched to c_j residents in $T_{i,j}$ then r_i must be matched to a hospital in $S_{i,j}$.

2. if r_i is not matched to some hospital in $S_{i,j}$ then h_j must be matched to c_j residents in $T_{i,j}$.

The objective function, which is a summation of all the binary variables, seeks to maximize the size of the matching.

$$\max \sum_{i=1}^{n_1} \sum_{h_j \in P(r_i)} x_{i,j}$$

subject to

1. $\sum_{h_j \in P(r_i)} x_{i,j} \leq 1 \qquad (1 \leq i \leq n_1)$
2. $\sum_{r_i \in P(h_j)} x_{i,j} \leq c_j \qquad (1 \leq j \leq n_2)$
3. $c_j \left(1 - \sum_{h_q \in S_{i,j}} x_{i,q}\right) - \sum_{r_p \in T_{i,j}} x_{p,j} \leq 0 \qquad (1 \leq i \leq n_1, h_j \in P(r_i))$

$x_{i,j} \in \{0, 1\}$

Figure 3.1: `Model1`: A HRT IP model

Theorem 3.2.1. *Given an* HRT *instance I modelled as an IP using* `Model1`*, a feasible solution to* `Model1` *produces a weakly stable matching in I. Conversely a weakly stable matching in I corresponds to a feasible solution to* `Model1`*.*

Proof. Assume that the IP model has a feasible solution $\mathbf{x}$. Let

$$M = \{(r_i, h_j) \in R \times H : r_i \in R \wedge h_j \in P(r_i) \wedge x_{i,j} = 1\}.$$

be the matching in I generated from $\mathbf{x}$. By Constraints 1 and 2, M is a matching: each resident is matched to at most one hospital, residents are only ever assigned to acceptable hospitals, and hospitals do not exceed their capacities. Constraint 3 ensures that the matching is weakly stable: if M was not a weakly stable matching then some pair (r_i, h_j) would block M. Thus r_i prefers h_j to $M(r_i)$ or r_i is unmatched. In both cases, $\sum_{h_q \in S_{i,j}} x_{i,q} = 0$, meaning the first term in constraint 3 would yield a value of c_j. Also h_j is either under-subscribed or h_j strictly prefers r_i to one of the residents in $M(h_j)$. In both cases, the second term of Constraint 3 would be less than c_j thus the IP solution would be infeasible, a contradiction.

Conversely, assume that M is a weakly stable matching in I. We can show that M corresponds to a feasible solution to `Model1`. Initially for all i $(1 \leq i \leq n_1)$, j $(1 \leq j \leq n_2)$, let $x_{i,j} = 0$. If $(r_i, h_j) \in M$ then set $x_{i,j} = 1$. Constraints 1 and 2 are satisfied as M is a matching. For Constraint 3 not to be satisfied the first term of the left-hand-side

must be greater than the second. Thus for some i ($1 \leq i \leq n_1$) and j ($h_j \in P(r_i)$), r_i is either unmatched or strictly prefers h_j to $M(r_i)$, thus making the first term result in a value of c_j, as the first term can only have a value of 0 or c_j. The second term of the left-hand-side must then be less than c_j. This means that the number of residents r_p assigned to h_j, such that h_j either prefers r_p to r_i or is indifferent between them, is less than c_j. Thus (r_i, h_j) would block M in I, a contradiction. □

3.3 Implementing the model

In this section we describe some techniques used to reduce the size of the HRT IP model generated and improve the performance of the IP solver. We also specify how the implementation was tested for correctness.

3.3.1 Reducing the model size

Techniques were described in [60] for removing acceptable pairs that cannot be part of any stable matching from HRT instances with ties on one side of the preference lists only. The `Hospitals-offer` and `Residents-apply` algorithms described therein identify pairs that cannot be involved in any stable matching, nor form a blocking pair with respect to any stable matching, and remove them from the instance. This produces a reduced HRT instance that would yield fewer variables and constraints when modelled as an IP, thus speeding up the optimisation process. The original instance and the reduced instance have the same set of stable matchings. These techniques as described in [60] are only applicable to instances where ties exists on one side only. This is a natural restriction that can be found in practice where hospitals usually rank residents based on performance grades (which can easily be tied) with residents being required to provide a strict preference of a bounded length (say 5 hospitals). We extend the `Hospitals-offer` algorithm slightly to allow for ties on both sides of an HRT instance. However, as we shall see later in this section, extending the `Residents-apply` algorithm to allow for ties on both sides will significantly reduce its ability to cut down the size of the HRT instance.

To help explain the `Hospitals-offer` and `Residents-apply` procedures, we first define some useful terms. We begin with the `Hospitals-offer` procedure shown in Algorithm 3.1. The process involves hospitals offering positions to residents, who form provisional arrangements with the hospitals. We define an *active tie* T_j on a hospital h_j's preference list as a tie containing one or more residents that immediately follows the least preferred resident currently assigned to h_j. Quantity $t_j = |T_j|$ may be 0 if such a tie does not

Algorithm 3.1 `Hospitals-offer`

```
1: while (there is a hospital h_j such that v_j ≥ t_j > 0) do
2:    for each resident r_i ∈ T_j do
3:       if r_i is already assigned, say to h_k then
4:          unassign r_i;
5:          increment v_k;
6:       assign r_i to h_j;
7:       decrement v_j;
8:       for each each hospital h_l that is a strict successor of h_j on r_i's list do
9:          delete the pair (r_i, h_l) from the preference lists;
```

Algorithm 3.2 `Residents-apply`

```
1: while (some resident r_i is free and has a nonempty list) do
2:    h_j = first hospital on r_i's list;
3:    assign r_i to h_j;
4:    increment a_j;
5:    if a_j ≥ c_j then
6:       let r_k be one of h_j's c_j^th-choice assignees;
7:       for each each strict successor r_l of r_k in h_j's list do
8:          if r_l is assigned to h_j then
9:             break the assignment;
10:            decrement a_j;
11:         delete the pair (r_l, h_j) from the preference lists;
```

exist. Also the value of t_j may change during the execution of the `Hospitals-offer` algorithm due to potential deletions of pairs from the problem instance. We denote by v_j the number of vacancies in h_j; this is the difference between the capacity c_j and the number of residents currently assigned to it. Hospital h_j is full when $v_j = 0$ and undersubscribed when $v_j > 0$. For each hospital that has a number of vacancies greater than or equal to the size of the current active tie, the residents in the active tie are assigned to that hospital. If these residents were previously assigned to other hospitals, those assignments are broken. For each of these residents $r_i \in T_j$, this is followed by the removal, from r_i's preference list, of all the hospitals that are *strict* successors to h_j. The process terminates when each hospital has an active tie of length 0, or its number of vacancies is less than the size of its active tie.

The `Residents-apply` procedure shown in Algorithm 3.2, involves residents successfully applying to hospitals on their preference list in preference order, again forming provisional assignments with the hospitals. We define a_j to be the the number of assignees of a hospital h_j. Each unassigned resident (say r_i) applies to the first hospital on his/her preference list (say h_j). Resident r_i is then temporarily assigned to h_j and a_j is incremented. This assignment may make h_j full ($a_j = c_j$) or oversubscribed ($a_j > c_j$). In both cases, each *strict* successor r_l of h_j's c_j^{th} assignee is then removed from h_j's preference list and h_j is removed from r_l's preference list. This deletion may

cause some resident r_k previously assigned to h_j to be free again (if this happens a_j is decremented). That resident can then apply to the next hospital on his/her preference list. The process will continue while some resident is free and has a hospital on his/her preference list. If ties were allowed to exist on the residents' preference list then other hospitals may be tied with h_j and so the deletions described cannot be carried out until all such hospitals are either full or oversubscribed. This extra condition will reduce the amount of deletions and so limit the effect of the algorithm.

3.3.2 Improving optimisation performance

A number of steps can be taken to improve the optimisation performance of the models. These generally involve heuristics and techniques that limit the number of nodes of the branch-and-cut tree that need to be evaluated in order to arrive at an optimal solution. Some of the steps relevant to our model are described below.

Placing a lower bound on the objective function can speed up the optimisation process of the IP solver. This can be added as an extra constraint to the model. This extra constraint will not affect the structure of the HRT model or nullify the proofs already provided. It simply allows the IP solver to ignore solutions that have sizes less than the bound provided. This lower bound can be calculated by firstly solving the HRT instance using any of the approximation algorithms described in [60] (for example Király's $O(n^2)$ approximation algorithm for MAX SMTI [75]) and obtaining the size of the maximum stable matching found. The objective function is then used as the linear expression of the new constraint added. An upper bound for the problem, which corresponds to the size of a maximum matching in the underlying bipartite graph instance can also be set. This leads to the introduction of the following two constraints:

$$\sum_{i=1}^{n_1} \sum_{h_j \in P(r_i)} \geq \text{lower bound} \qquad \sum_{i=1}^{n_1} \sum_{h_j \in P(r_i)} \leq \text{upper bound}$$

Another technique that can help to speed up the optimisation process is to provide the solver with an initial solution. This cuts down on the time needed for the solver to initially solve the LP and start finding an IP solution. This initial solution to the HRT model can once again be obtained by solving the instance using any of the algorithms described in [60] and obtaining the maximum stable matching.

3.3.3 Testing the implementation

Although the theoretical model has been proven to be correct in Theorem 3.2.1, it is still important to verify the correctness of the implementation. The system was tested to ensure a high degree of confidence in the results of the experimental evaluation, presented later in this chapter. The correctness of the pre-processing steps and the IP solution were evaluated by generating multiple instances (100,000) of various sizes (with up to 400 residents) and testing the stability and size of the resulting matching against both the original and the trimmed problem instances. The matchings generated can also be tested to ensure that the hospitals do not exceed their capacities, residents are assigned to at most one hospital, and assignments are consistent (meaning that if r_i is assigned to h_j then h_j is also assigned to r_i). For all the instances tested, the solver produced stable matchings.

Stability is just one property of the resulting matching that needs to be tested. Another important property is optimality. One method of testing the optimality of the stable matchings produced would be to compare the sizes of optimal solutions generated by various models on the same problem instance. Although we have presented only one IP model, HRT can be modelled as an integer programmming problem in various ways. Implementing these different models and comparing their outputs gives some indication of the correctness of the implementation. This test is not fool-proof as errors in the implementations may be duplicated across the models. An alternative is to test for optimality using a more exhaustive approach.

Another method to verify that the stable matchings generated by the IP model are optimal, is to generate all stable matchings in the HRT instance. The largest stable matching found would then be compared to the optimal solution obtained by the solver. If the sizes of both matchings are the same, we confirm that the IP solver produced an optimal solution to that instance. A difference would indicate a fault in one or both of the methods. The larger the number of instances tested and the bigger the size of instances, the more confident we will be of the IP implementation. The generation of all stable matchings was done by finding all matchings in the underlying bipartite graph, testing for the stability of each matching and keeping a record of the largest stable matching found.

The `Generate-max-hrt` algorithm (shown in Algorithm 3.3) is a brute force technique used to find a maximum weakly stable matching given a HRT instance I. We define a global variable M_{opt} to represent the largest weakly stable matching in I discovered. Next we make an initial call to the recursive `Choose` algorithm. The `Choose` algorithm requires an integer i $(1 \leq i \leq n_1)$ and a matching M. In each call to the `Choose` algorithm, we try to add a pair involving resident r_i and one of the hospitals h_j on her

Algorithm 3.3 `Generate-max-hrt`

Require: a HRT instance I;
largest stable matching $M_{opt} = \emptyset$;
`choose`$(1, \emptyset)$;
return M_{opt};

Algorithm 3.4 `Choose`

Require: an integer i and a matching M;
if $i > n_1$ **then**
  if $|M| > |M_{opt}|$ and M is weakly stable **then**
    $M_{opt} = M$;
else
  for each $h_j \in P(r_i)$ **do**
    if r_i is unmatched in M and h_j is undersubscribed in M **then**
      $M = M \cup \{(r_i, h_j)\}$;
      `choose`$(i + 1, M)$;
      $M = M \backslash \{(r_i, h_j)\}$;
  `choose`$(i + 1, M)$;

preference list if both are available (i.e. r_i is unmatched and h_j is undersubscribed). For the initial call to `Choose` the values of i and M provided are 1 and $\emptyset$ respectively. In the `Choose` algorithm we consider each hospital $h_j \in P(r_i)$. If r_i is unmatched in M and h_j is undersubscribed in M we add (r_i, h_j) to M and call `Choose` with values $i+1$ and M passed in. After the recursive call we remove (r_i, h_j) from M. Finally after considering all the hospitals in $P(r_i)$ we then consider the case where r_i is unmatched by calling `Choose` with values $i + 1$ and M passed in. The `Choose` algorithm exits when all residents have been considered. Before exiting, if M is larger than M_{opt} and is weakly stable in I, we replace M_{opt} with M. The main `Generate-max-hrt` algorithm terminates when the initial call to `Choose` returns and M_{opt} is returned.

Due to the brute-force nature of this technique we could only generate and test to optimality 100 instances each with number of residents $n_1 \in \{8, 9, 10, 11, 12\}$. For all the instances tested, the solver produced optimal stable matchings.

3.4 Empirical evaluation

An empirical evaluation of the IP model was carried out. Large numbers of random instances of HRT were generated by varying certain parameters relating to the construction of an instance and passed on to the CPLEX 12.5.1 IP solver. Data from past SFAS matching runs were also modelled and solved. This section discusses the methodology used and some of the results obtained. Experiments were carried out on a Linux machine with 8 Intel(R) Xeon(R) CPUs at 2.5GHz and 32GB RAM.

t_d	$n_1 = 200$	$n_1 = 250$	$n_1 = 300$
75%	100.00%	100.00%	99.85%
80%	99.98%	99.88%	99.39%
85%	99.90%	99.29%	97.76%
90%	99.70%	99.28%	98.60%
95%	99.99%	100.00%	100.00%

Table 3.1: Percentage solvable instances (100% for omitted t_d values)

3.4.1 Using random instances

Random HRT problem instances were generated using Java's random number generator. The instances consist of n_1 residents, n_2 hospitals and C posts where n_1, n_2 and C can be varied. The hospital posts were randomly distributed amongst the hospitals. Other properties of the generated instance that can be varied include the lengths of residents' preference lists as well as a measure of the density t_d of ties present in the preference lists. The tie density t_d ($0 \leq t_d \leq 1$) of the preference lists is the probability that some agent is tied to its successor agent in a given preference list. At $t_d = 1$ each preference list would comprise a single tie while at $t_d = 0$ no tie would exist in the preference lists of the agents thus yielding an HR instance.

3.4.1.1 Varying tie density

Since ties cause the sizes of stable matchings to vary, an obvious question to investigate is how the variation in tie density affects the runtime of the IP model and the size of the maximum stable matchings found. These values were measured for multiple instances of MAX HRT while varying the tie density t_d of hospitals' preference lists. There were no ties in the residents' preference lists. This was done for increasing sizes ($n_1 = 200, 250, 300$) of the problem instance with each resident's preference list being strictly ordered and of length 5. A total of $10,000$ instances were randomly generated for each tie density value (starting at $t_d = 0\%$ to $t_d = 100\%$ with an interval of 5%) and instance size. For each instance $C = n_1$ and $n_2 = \lfloor 0.07 \times n_1 \rfloor$.

To avoid extreme outliers skewing the mean time measures, we define what we regard as a reasonable solution time (namely 300 seconds) and abandon search if the solver exceeds this cut-off time. For most tie densities this cut-off was not exceeded for the values of n_1 and t_d considered. Table 3.1 shows the percentage of instances that were solved before the cut-off was exceeded (omitted t_d values were 100% solvable).

From Figures 3.2 and 3.3 we see that the mean and median runtime (taken over the instances that did not timeout) remain significantly low for instances with $t_d < 60\%$ but then gradually increase until they reach their peaks (in the region of $80\% - 90\%$)

before falling as the tie density approaches 100%. From a theoretical perspective, it is known that the problem is polynomially solvable when the tie density is at both 0% and 100% and it is easy to see how the IP solver will find these cases trivial. As the tie density increases the number of stable matchings that the instance is likely to admit also increases, explaining the observed increase in the runtime. The `Hospitals-offer` and `Residents-apply` algorithms used to trim the instance also play their part in this trend with limited trimming done for higher tie densities.

Figure 3.4 shows the variation in optimal values with tie density for $n_1 = 300$ (again taken over the instances that did not timeout). We observe an increase in the average size of maximum stable matchings as the tie density increases. This is in line with the idea that the stability requirement restricts the size of stable matchings and increasing tie density can be viewed as relaxing stability requirements.

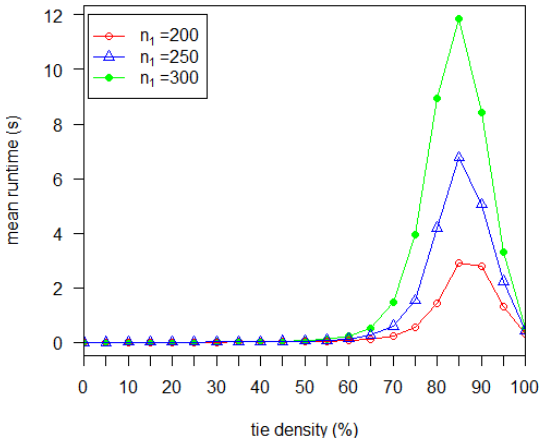

Figure 3.2: Mean runtime vs t_d

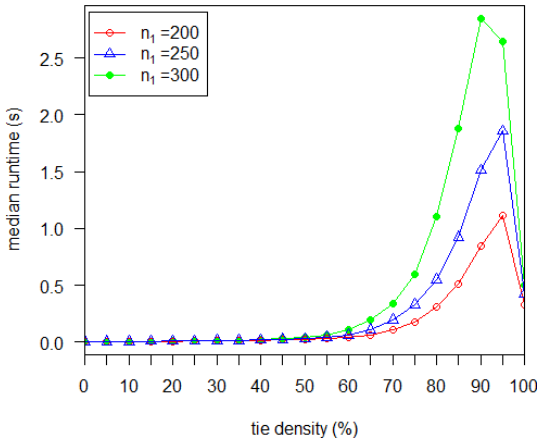

Figure 3.3: Median runtime vs t_d

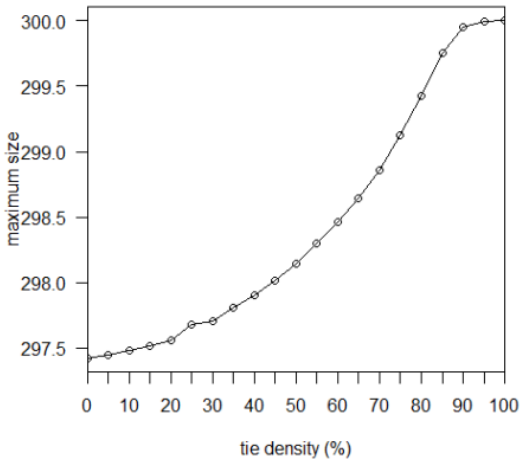

Figure 3.4: $|M|$ vs t_d for $n_1 = 300$

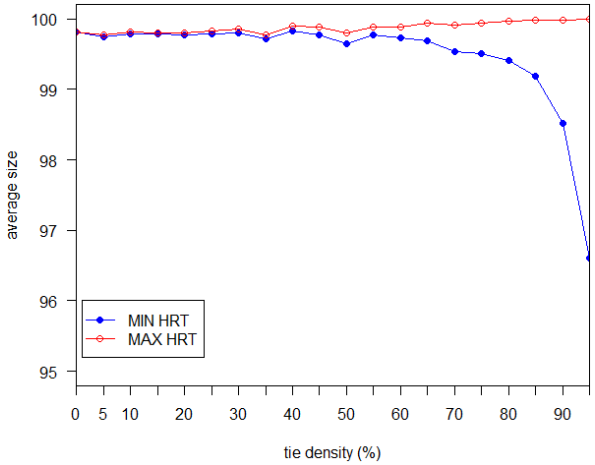

Figure 3.5: Range vs t_d

3.4.1.2 Comparing MAX HRT and MIN HRT

We have established that the presence of ties in the preference lists makes MAX HRT NP-hard. In Section 3.4.1.1 we observed that our IP model only begins to experience this computational difficulty in practice when tie densities exceed the 60% region. Given this observation, a natural trend to investigate is how the *range* in the size of the stable matchings will vary with tie density. We define the range in this context as the difference between the sizes of a minimum and a maximum stable matching. It is fair to assume that as the range gets wider there will be an increase in the number of stable matchings thus making the solver do more work in order to find an optimal solution. As in the case of the experiment in Section 3.4.1.1 we generated multiple (100) instances of HRT for each tie density value (in the range $0\%, 5\%, 10\%, ..., 95\%$) and solved both MAX HRT and MIN HRT IP models. Once again, residents' preference lists were kept strict and at a length of 5. Also for each instance $C = n_1$ and $n_2 = \lfloor 0.07 \times n_1 \rfloor$. All instances were solved to optimality in this experiment.

Figure 3.5 shows the results obtained from running the experiment on instances of size 100. We observe that the average range begins to increase significantly as the tie density increases beyond 60%. This is in line with the results presented in Figures 3.2 and 3.3.

3.4.1.3 Increasing instance size

The execution time for solving multiple MAX HRT instances of increasing sizes via IP models was evaluated. The tie density t_d and preference list lengths were kept constant. This provided an estimate of problem sizes for which the IP model can be of practical value. The tie densities of the hospitals' preference lists were set to 0.85 on all instances. On the bases of the results from Figures 3.2 and 3.3, we estimate that instances with tie density in the region 0.7–0.95 will take longer to be solved than for other density values. There were no ties in the residents' preference lists. The instance size n_1 was increased by 100 starting at $n_1 = 100$. A total of 100 instances for each value of n_1 was generated. The number of hospitals n_2 in each instance was set to $\lfloor 0.07 \times n_1 \rfloor$. Each resident has a preference list of 5 hospitals with each hospitals' preference list length being determined by the frequency of its occurrence within the residents' lists. A cut-off value of 300 seconds was also set with the percentage of solvable instances for 400, 500 and 600 instances being 95%, 93% and 92% respectively.

Figure 3.6 shows how the mean and median runtimes rise as n_1 increases. Although the trend for the mean runtime seems to suggest that it grows exponentially with time, we observe that the IP is still scalable for considerably large datasets. We also observe

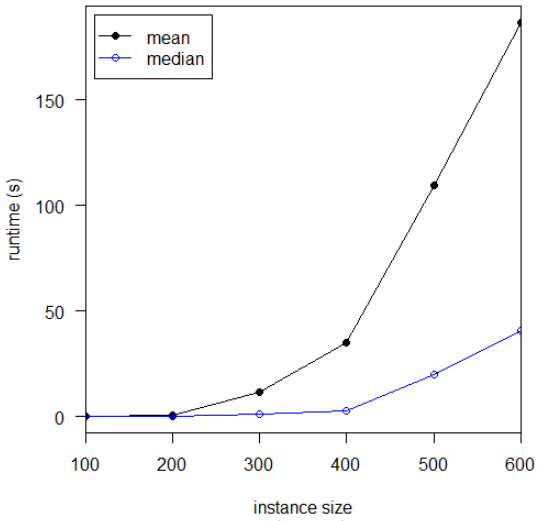

Figure 3.6: Mean and median runtime vs instance size

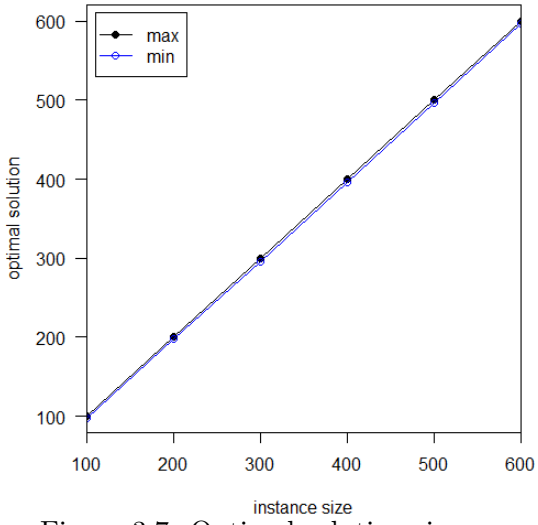

Figure 3.7: Optimal solution size vs instance size

the considerable difference between the mean and median runtime as the instance size grows. This suggests that the existence of challenging instances becomes more probable, leading to a more rapid increase in the mean runtime compared to the median runtime. Figure 3.7 shows the difference between the mean and min optimal solutions found. It is interesting to observe that optimal solutions tend to match almost every resident given the instance generation parameters used.

3.4.2 Using real instances

Another question worth asking is whether the IP model can handle instance sizes found in real-world applications. In [60], various approximation algorithms and heuristics were implemented and tested on real datasets from the SFAS matching scheme for 2006, 2007 and 2008, where the residents' preferences are strictly ordered and of length 6 with ties existing in the hospitals' preference lists. With the IP model, it is now possible to trim the instances using the techniques mentioned in Section 3.3.2, generate an optimal solution and compare the results obtained with those reported in [60]. Results from these tests showed that, while some algorithms did marginally better than others, all the algorithms developed generated relatively large stable matchings with respect to the optimal values.

year	n_1	n_2	t_d	time (sec) for IP model	$\|M\|$	$\|M'\|$ from [60]
2006	759	53	92%	92.96	758	754
2007	781	53	76%	21.78	746	744
2008	748	52	81%	75.50	709	705

Table 3.2: SFAS IP Results

Table 3.2 shows this comparison. Let M' denote the largest stable matching found over all the algorithms tested in [60]. We measured the tie density by dividing the sum of the lengths of all ties by the sum of the lengths of all the hospitals' preference lists. We observe that the tie densities lie within the range $0.75 - 0.95$ meaning that these instances are likely to be harder to solve. The measured runtimes confirm this.

3.5 Open problems

We identify below some important open problem relating to IP models for MAX HRT.

1. Although the performance of the IP model presented in this chapter was very good for instance sizes that can be expected in many practical applications, the scalability of the approach still needs to be improved. This requirement leads to the possibility of adopting the technique called *column generation* [18]. A column generation approach was described for the kidney exchange problem with a bounded cycle length [6] and results indicated considerable improvements in performance. It is expected that adopting this technique would improve the performance of our HRT IP model.

2. Given a sub-problem with a number of variables with non-integer values, the choice of which variable is selected to decompose the problem into two sub-problems can have an effect on the performance of the branch-and-cut algorithm. Perhaps special knowledge of matching problems can help with this decision. Maybe variables that represent entries on shorter preference lists should be given higher priorities or variables that represent entries that are involved in ties. Empirical results may shed more light on which variable-ordering heuristics are beneficial in practice.

3. With a number of nodes in the search tree being eligible for consideration at any time during the execution of the branch-and-cut algorithm, the choice of how to prioritise the selection of these nodes might lead to changes in the performance of the algorithm. We may choose to let the nodes be selected randomly or we may deliberately specify a *depth-first* or *best-node-first* approach.

   In a depth-first approach, priority is given to newly-discovered nodes, thus descending further down the search tree, from a node to its immediate descendant, would quickly yield the first feasible solution which can then serve as a good lower bound. This would be a good approach for problems where finding a feasible solution is generally hard in practice and the model is not seeded with an

initial solution. In our models however, we already have reasonably good lower bounds provided by the initial solutions used to seed the model. In such a case, it might be advantageous to branch using the best-node-first approach. Priority is given to the node with the largest upper bound as this will ensure that nodes with upper bounds less than the optimal value will never be considered.

Chapter 4

Socially Stable Matchings in the Hospitals/Residents Problem

4.1 Introduction

As discussed in Chapter 1, a number of scenarios exist where the adoption of weaker notions of stability may be appropriate. The typical motivation for relaxing the stability requirement is the potential to discover larger matchings while still reducing the likelihood that the matchings would unravel due to the presence of blocking pairs. In Chapter 3 we investigated one of such cases where the preference lists of the agents involved contain ties. In this chapter we consider another scenario where the (social) relationship between agents may have an impact on their ability (or inability) to form blocking pairs with respect to a matching. In Section 2.5 we introduced this idea (which is called local stability in the literature) and presented a review of the current literature in the area. In this chapter we extend this notion of local stability (which we call *social stability*) and the problem of finding a socially stable matching in the Hospitals/Residents problem. We formally define the problem (which we call the *Hospitals/Residents problem under Social Stability* (HRSS)) and present algorithmic results for HRSS and its variants.

In Section 4.2, we present some preliminary definitions and observations. In Section 4.3 we give a reduction from HRSS to the HR+SN problem (defined in Section 2.5). In Section 4.4 we show that MAX HRSS, the problem of finding a maximum socially stable matching given an HRSS instance I, is NP-hard even under certain restrictions on the lengths of the preference lists. This result holds even if I is an instance of SMISS (the 1-1 special case of HRSS). In Section 4.5, we provide approximability results for MAX HRSS including a 3/2-approximation algorithm for the problem. Then in Section 4.6 we present polynomial-time algorithms for three special cases of MAX HRSS where the

lengths of the preference lists or the numbers of acquainted or unacquainted pairs may be bounded. Section 4.7 gives results obtained from an empirical evaluation of the 3/2-approximation algorithm for MAX HRSS, applied to randomly-generated instances of HRSS, where the objective is to measure how the mean execution times and matching sizes change as we vary the instance size and the social network density. The chapter ends with a brief conclusion in Section 4.8.

4.2 Preliminary definitions and results

Given its importance to HRSS, we begin by reminding the reader of the definition of HR (as presented in Section 2.3). An instance I of the *Hospitals/Residents problem* (HR), as defined in [29], contains a set $R = \{r_1, r_2, ..., r_{n_1}\}$ of residents and a set $H = \{h_1, h_2, ..., h_{n_2}\}$ of hospitals. Each resident $r_i \in R$ ranks a subset of H in strict order of preference; whilst each hospital $h_j \in H$ ranks a subset of R, consisting of those residents who ranked h_j, in strict order of preference. Each hospital h_j also has a *capacity* $c_j \in \mathbb{Z}^+$ indicating the maximum number of residents that can be assigned to it. A pair (r_i, h_j) is called an *acceptable pair* if h_j appears in r_i's preference list. We denote by $\mathcal{A}$ the set of all acceptable pairs. A *matching* M is a set of acceptable pairs such that each resident is assigned to at most one hospital and the number of residents assigned to each hospital does not exceed its capacity. If r_i is matched in M, we denote the hospital assigned to resident r_i in M by $M(r_i)$. We denote the set of residents assigned to hospital h_j in M as $M(h_j)$. A resident r_i is *unmatched* in M if no pair in M contains r_i. A hospital h_j is *undersubscribed* in M if $|M(h_j)| < c_j$. A pair (r_i, h_j) is said to *block* a matching M, or form a *blocking pair* with respect to M, in the classical sense, if (i) r_i is unmatched in M or prefers h_j to $M(r_i)$ and (ii) h_j is undersubscribed in M or prefers r_i to some resident in $M(h_j)$. A matching that admits no blocking pair is said to be *stable*.

We define an instance (I, G) of the *Hospitals/Residents Problem under Social Stability* (HRSS) as consisting of an HR instance I (as defined above) and a bipartite graph $G = (R \cup H, A)$, where $A \subseteq \mathcal{A}$. A pair (r_i, h_j) belongs to A if and only if r_i has social ties with h_j (i.e., r_i is acquainted with h_j in some way). We call (r_i, h_j) an *acquainted pair*. We also define the set of *unacquainted* pairs (which cannot block any matching) to be $U = \mathcal{A} \backslash A$. A pair (r_i, h_j) *socially blocks* a matching M, or forms a *social blocking pair* with respect to M, if (r_i, h_j) blocks M in the classical sense in the underlying HR instance I and $(r_i, h_j) \in A$. A matching M is said to be *socially stable* if there exists no social blocking pair with respect to M. If we restrict the hospitals' capacities to 1, we obtain the *Stable Marriage problem with Incomplete lists under Social Stability* (SMISS),

men's preferences	women's preferences
$m_1 : w_1$	$w_1 : m_2 \quad m_1$
$m_2 : w_1 \quad w_2$	$w_2 : m_2$

social network graph G

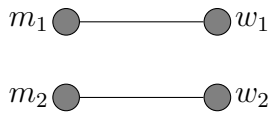

Figure 4.1: An SMISS instance (I, G) consisting of a HR instance I and a social network G

and refer to the agents as *men* $\mathcal{U} = \{m_1, \ldots, m_{n_1}\}$ and *women* $\mathcal{W} = \{w_1, \ldots, w_{n_2}\}$.

Clearly every instance of HRSS admits a socially stable matching. This is because the underlying HR instance is bound to admit a stable matching [29] which is also socially stable. However socially stable matchings could be larger than stable matchings. Consider the SMISS instance (I, G) shown in Figure 4.1. Matchings $M_1 = \{(m_1, w_1), (m_2, w_2)\}$ and $M_2 = \{(m_2, w_1)\}$ are both socially stable in (I, G) and M_2 is the unique stable matching. Thus an instance of SMISS (and hence HRSS) can admit a socially stable matching that is twice the size of a stable matching. Clearly the instance shown in Figure 4.1 can be replicated to give an arbitrarily large SMISS instance with a socially stable matching that is twice the size of a stable matching. This observation, together with the fact that in many applications we seek to match as many agents as possible, motivates MAX HRSS.

4.3 Reduction from HRSS to HR+SN

As defined in [11, 22], an instance (I, G') of the HR+SN problem involves a Hospitals / Residents instance I, defined in [29], containing a set $R = \{r_1, r_2, ..., r_{n_1}\}$ of residents, a set $H = \{h_1, h_2, ..., h_{n_2}\}$ of hospitals, and a graph G describing the social network (SN) of the residents. In the graph $G = (V, E)$, $V = R$ and an edge $\{r_i, r_k\}$ belongs to E if and only if r_i and r_k have *social ties*. A pair (r_i, h_j) is a *local blocking pair* with respect to a matching M, or *locally blocks* M, if (r_i, h_j) blocks M in the classical sense and there is some resident r_k such that $\{r_i, r_k\} \in E$ and $r_k \in M(h_j)$. A matching M is said to be *locally stable* if there exists no local blocking pair with respect to M. In the HR+SN (respectively HRSS) context we refer to a *resident-complete* locally (respectively socially) stable matching as one in which all the residents are matched.

In this section we show the close relationship between the HRSS and HR+SN problems. Consider an instance (I, G) of HRSS where I is the underlying HR instance

and G is the social network graph. I involves a set of residents $R_0 = \{r_1, r_2, ..., r_{n_1}\}$ and a set of hospitals $H_0 = \{h_1, h_2, ..., h_{n_2}\}$. We construct an instance (I', G') of HR+SN from (I, G) as follows: let I' consist of a set of residents $R = R_0 \cup R_1$ where $R_1 = \{r_{n_1+1}, r_{n_1+2}, ..., r_{n_1+n_2}\}$. Every resident $r_{n_1+j} \in R_1$ has a single entry h_j in his preference list. Every resident $r_i \in R_0$ has the same preference list in I' as he has in I. Let I' also involve a set of hospitals H, where $H = H_0$ and every hospital $h_j \in H$ has resident r_{n_1+j} as the first entry in its preference list and has capacity $c'_j = c_j + 1$. Hospital h_j's preference list in I is then appended to r_{n_i+j} to yield h_j's preference list in I'. To construct G', let the vertices in G' correspond to the residents in R and add edge $\{r_i, r_{n_1+j}\}$ to G' if and only if $(r_i, h_j) \in A$, where $A = E(G)$ is the set of acquainted pairs in (I, G).

Theorem 4.3.1. *If M is a socially stable matching in (I, G), then $M' = M \cup \{(r_{n_1+j}, h_j) : r_{n_1+j} \in R_1\}$ is a locally stable matching in (I', G'). Conversely if M' is a resident-complete locally stable matching in (I', G') then $M = M' \backslash \{(r_{n_1+j}, h_j) : r_{n_1+j} \in R_1\}$ is a resident-complete socially stable matching in (I, G).*

Proof. Suppose M is socially stable in (I, G). Then no (classical) blocking pair with respect to M in I is contained in G. Let $M' = M \cup \{(r_{n_1+j}, h_j) : r_{n_1+j} \in R_1\}$. If some pair (r_i, h_k) locally blocks M' in (I', G') then (i) (r_i, h_k) must be a blocking pair with respect to M' in I', and (ii) $\{r_i, r'_i\} \in E$ for some $r'_i \in M'(h_k)$. By construction, for every edge in E, one resident is in R_0 and the other in R_1. If $r_i \in R_1$ then r_i cannot form any blocking pair with respect to M' as he is matched to his only choice. If $r_i \in R_0$ then $r'_i = r_{n_1+j} \in R_1$ for some j $(1 \leq j \leq n_2)$, and $h_k = M'(r_{n_1+j}) = h_j$. Thus $(r_i, h_k) \in A$. By the construction of the preference lists in (I', G'), as (r_i, h_k) is a (classical) blocking pair of M' in I', (r_i, h_k) is also a (classical) blocking pair of M in I. Hence (r_i, h_k) socially blocks M in (I, G), a contradiction.

Conversely suppose M' is a resident-complete locally stable matching in (I', G'). Then there is no blocking pair (r_i, h_k) of M' in I' such that $\{r_i, r'_i\} \in E$ for some $r'_i \in R$ where $r'_i \in M'(h_k)$. Let $M = M' \backslash \{(r_{n_1+j}, h_j) : r_{n_1+j} \in R_1\}$. Clearly M is a resident-complete matching in I. If some pair (r_i, h_k) socially blocks M in (I, G), (i) (r_i, h_k) must block M in I (and thus M' in I') and (ii) $(r_i, h_k) \in A$. By construction, if $(r_i, h_k) \in A$, then $\{r_i, r'_i\} \in E$ where $r_i \in R_0$, $r'_i = r_{n_1+j} \in R_1$ and $h = h_j$. But as M' is resident-complete, $(r_{n_1+j}, h_j) \in M'$ for each j $(1 \leq j \leq n_2)$. Thus $r'_i = r_{n_1+j} \in M'(h_k)$, a contradiction to the initial assumption that M' is locally stable in (I', G'). □

Although the converse statement in Theorem 4.3.1 places a severe restriction on M' (it must be a resident-complete locally stable matching in the HR+SN instance (I', G')), it can be relaxed slightly to the case that M' is any locally stable matching in which all

the residents $r_{n_1+j} \in R_1$ are matched. It remains to be shown that a reduction exists from HRSS to HR+SN that does not place such a restriction on M'.

4.4 Hardness of MAX SMISS

We now show that MAX SMISS, the problem of finding a maximum socially stable matching given an SMISS instance is NP-hard. Indeed we prove NP-completeness for COM SMISS, the problem of deciding whether there exists a *complete* socially stable matching (i.e., a socially stable matching in which all men and women are matched) in an instance of SMISS. It was shown in [86] that COM SMTI, the problem of deciding whether a complete stable matching exists in an instance of SMTI, is NP-complete even if the ties occur in the men's lists only and each tie occurs at the tail of some list. An SMTI instance I satisfying these restrictions can be reduced to an SMISS instance (I', G) in polynomial time such that a matching M is a complete stable matching in I if and only if M is a complete socially stable matching in (I', G). These observations form the basis of the proof of the following result.

Theorem 4.4.1. COM SMISS *is NP-complete.*

Proof. It is obvious that COM SMISS is in NP. Consider an instance I of SMTI where the ties occur only on the men's preference lists and each man has one tie which occurs at the end of the list (a tie may be of length 1 for this purpose). We define $t(m_i)$ as the set of women contained in the tie in man m_i's preference list. We can construct an instance (I', G) of SMISS such that I' is the SMI instance formed by breaking the ties in I in an arbitrary manner. Let $G = (\mathcal{U} \cup \mathcal{W}, A)$, where $\mathcal{U}$ and $\mathcal{W}$ are the sets of all men and women in I respectively, $U = \bigcup_{m_i \in \mathcal{U}} \bigcup_{w_j \in t(m_i)} (m_i, w_j)$ and $A = \mathcal{A} \backslash U$. We claim that a matching M is a complete stable matching in I if and only if M is a complete socially stable matching in (I', G).

Suppose M is a complete stable matching in I. Suppose also that M is not socially stable in (I', G). Then there exists some pair $(m_i, w_j) \in A$ that socially blocks M in (I', G). Since $(m_i, w_j) \in A$, $w_j \notin t(m_i)$. Thus m_i prefers w_j to $M(m_i)$ in I. Also w_j prefers m_i to $M(w_j)$ since there are no ties in w_j's preference list. Thus (m_i, w_j) blocks M in I, a contradiction to our initial assumption.

Conversely, suppose M is a complete socially stable matching in (I', G). Suppose also that M is not stable in I. Then there exists some pair (m_i, w_j) that blocks M in I. If $M(m_i) \in t(m_i)$ then $w_j \notin t(m_i)$ so m_i prefers w_j to $M(m_i)$ in (I', G) and $(m_i, w_j) \in A$. Likewise if $M(m_i) \in t(m_i)$ then m_i prefers w_j to $M(m_i)$ in (I', G) and $(m_i, w_j) \in A$.

Woman w_j has the same preference list in I and (I', G). Thus (m_i, w_j) socially blocks M in (I', G), a contradiction to our initial assumption. □

As discussed in [61], some centralised matching schemes usually require the agents in one or more sets to have preference lists bounded in length by some small integer. For example, until recently, in the Scottish Foundation Allocation Scheme (the centralised clearinghouse for matching medical residents in Scotland) [52], medical graduates were required to rank only 6 hospitals in their preference lists. We denote by (p, q)-MAX HRSS the problem of finding a maximum socially stable matching in an HRSS instance where each resident is allowed to rank at most p hospitals and each hospital at most q residents. We set $p = \infty$ and $q = \infty$ to represent instances where the residents and hospitals respectively are allowed to have unbounded-length preference lists. Analogously we may obtain the definition of (p, q)-MAX SMISS and (p, q)-COM SMISS from MAX SMISS and COM SMISS respectively. It turns out that (p, q)-COM SMISS is NP-complete even for small values of p and q.

Theorem 4.4.2. $(3, 3)$-COM SMISS *is NP-complete.*

Proof. We prove this by inspecting the proof of the NP-completeness result described for the $(3, 3)$-COM SMTI problem by Irving et al. [61]. They showed that $(3, 3)$-COM SMTI, the problem of deciding whether a complete stable matching exists in an instance of SMTI where each preference list is of length at most 3, is NP-complete using a reduction from a variant of the SAT problem.

By inspecting the instance I of SMTI, constructed in the proof, we observe that all the ties appear on one side of the instance and occur at the ends of the preference lists. We have shown in Theorem 4.4.1 that an instance I of SMTI in this form can be reduced in polynomial time to an instance (I', G) of SMISS such that a matching M is a complete stable matching in I if and only if M is a complete socially stable matching in (I', G). We conclude that $(3, 3)$-COM SMISS is also NP-complete. □

4.5 Approximating MAX HRSS

As shown in Section 4.4, MAX HRSS is NP-hard. In order to deal with this hardness, polynomial-time approximation algorithms can be developed for MAX HRSS. In this section we present a 3/2-approximation algorithm for MAX HRSS. We also show a $21/19 - \varepsilon$ lower bound for the approximability of this problem, for any $\varepsilon > 0$, assuming P≠NP. The lower bounds hold even for MAX SMISS.

4.5.1 Approximation results

For the upper bound for MAX HRSS, we observe that a technique known as *cloning* has been described in the literature [38, 106], which may be used to convert an HR instance I into an instance I' of SMI in polynomial time, such that there is a bijective function between the sets of stable matchings in I and I'. A similar technique can be used to convert an HRSS instance to an SMISS instance in polynomial time.

Let (I, G) be an instance of HRSS where $R = \{r_1, r_2, ..., r_{n_1}\}$ is the set of residents and $H = \{h_1, h_2, ..., h_{n_2}\}$ is the set of hospitals. Let c_j be the capacity of hospital $h_j \in H$. We can construct an instance (I', G') of SMISS as follows. Each resident in (I, G) corresponds to a man in (I', G'). Each hospital $h_j \in H$ gives rise to c_j women in (I', G'), denoted by $h_{j,1}, h_{j,2}, ..., h_{j,c_j}$, each of whom has the same preference list as h_j in (I', G') but with a capacity of 1. Each man $r_i \in R$ starts off with the same preference list in (I', G') as he has in (I, G). We then replace each entry on his list by the c_j women $h_{j,1}, h_{j,2}, ..., h_{j,c_j}$ listed in strict order (increasing on second subscript). G' has vertex set $R \cup H'$, where $H' = \{h_{j,k} : h_j \in H \wedge 1 \leq k \leq c_j\}$, and edge set $A' = \{(r_i, h_{j,k}) : (r_i, h_j) \in A \wedge 1 \leq k \leq c_j\}$, with $A = E(G)$ denoting the set of acquainted pairs in (I, G).

Theorem 4.5.1. *Given an instance (I, G) of* HRSS*, we may construct in $O(n_1 + c_{max}m)$ time an instance (I', G') of* SMISS *such that a socially stable matching M in (I, G) can be transformed in $O(c_{max}m)$ time to a socially stable matching M' in (I', G') with $|M'| = |M|$ and conversely, where n_1 is the number of residents, c_{max} is the maximum hospital capacity and m is the number of acceptable resident-hospital pairs in I.*

Proof. Let (I, G) be an instance of HRSS and (I', G') be an instance of SMISS cloned from (I, G). Let M be a socially stable matching in (I, G). We form a matching M' in (I', G') as follows. For each $h_j \in H$, let $r_{j,1}, r_{j,2}, ...r_{j,x_j}$ be the set of residents assigned to h_j in M where $x_j \leq c_j$, and $k < l$ implies that h_j prefers $r_{j,k}$ to $r_{j,l}$. Add $(r_{j,k}, h_{j,k})$ to M' $(1 \leq k \leq x_j)$. Clearly M' is a matching in (I', G') such that $|M'| = |M|$, and it is straightforward to verify that M' is socially stable in (I', G').

Conversely let M' be a socially stable matching in (I', G'). We form a matching M in (I, G) as follows. For each $(r_i, h_{j,k}) \in M'$, add (r_i, h_j) to M. Clearly M is a socially stable matching in (I, G) such that $|M| = |M'|$.

The complexities stated arise from the fact that I' has $O(n_1 + C)$ agents and $O(c_{max}m)$ acceptable man-woman pairs, where C is the total capacity of the hospitals in I. □

Due to Theorem 4.5.1, an approximation algorithm α for MAX SMISS with performance guarantee c (for some constant $c > 0$) can be used to obtain an approximation for MAX

HRSS with the same performance guarantee. This can be done by cloning the HRSS instance (I, G) to form an SMISS instance (I', G') using the technique outlined above, applying α to (I', G') to obtain a matching M'. This matching can then be transformed to a matching M in (I, G) such that $|M| = |M'|$ (again as in the proof of Theorem 4.5.1). Our first upper bound for MAX HRSS is an immediate consequence of the fact that in an SMISS instance, any stable matching is at least half the size of a maximum socially stable matching, as we now demonstrate.

Proposition 4.5.2. MAX HRSS *is approximable within a factor of 2.*

Proof. Let M be a maximum socially stable matching given an instance (I, G) of SMISS and let M' be a stable matching in I. Thus M' is a maximal matching in the underlying bipartite graph G' in I. Hence $|M'| \geq \beta^+(G')/2$ where $\beta^+(G')$ is the size of a maximum matching in G' [77]. Also $\beta^+(G') \geq |M|$ and so $|M'| \geq |M|/2$. □

We now present a 3/2-approximation algorithm for MAX SMISS. The algorithm relies on the principles outlined in the 3/2-approximation algorithms for the general case of MAX HRT, the problem of finding a maximum cardinality stable matching given an instance of the HRT, as presented by Király [75] and McDermid [90]. Given an instance (I, G) of SMISS, the algorithm works by running a modified version of the extended Gale-Shapley algorithm [29] where unmatched men are given a chance to propose again by promoting them on all the preference lists on which they appear. Let A and U denote the sets of acquainted and unacquainted pairs in (I, G) respectively.

Consider a woman w_j in (I, G). We denote an *unacquainted man m_i on w_j's preference list* as one where $(m_i, w_j) \in U$. Similarly we denote an *acquainted man m_i on w_j's preference list* as one where $(m_i, w_j) \in A$. During the execution of the algorithm if a man runs out of women to propose to on his list for the first time, he is *promoted*, thus allowing him to propose to the remaining women on his list beginning from the first. A man can only be promoted once during the execution of the algorithm. If a promoted man still remains unmatched after proposing to all the women on his preference list, he is removed from the instance and will not be part of the final matching. For a man m_i, we denote $next(m_i)$ as the next woman on m_i's list succeeding the last woman to whom he proposed to or the first woman on m_i's list if he has been newly promoted or is proposing for the first time.

In the classical Gale-Shapley algorithm [29] a woman w_j prefers a man m_i to another m_k if $rank(w_j, m_i) < rank(w_j, m_k)$. We define a modified version of the extended Gale-Shapley algorithm [38], `Mod-exgs`, where a woman does not accept or reject proposals from men solely on the basis of their positions on her preference list, but also on the basis of their status as to whether they are acquainted or unacquainted men on her list

and whether they have been promoted. Given two men m_i and m_k on a woman w_j's preference list, we define the relations $\lhd_{w_j}$, $\lhd'_{w_j}$ and $\prec_{w_j}$ as follows.

Definition 4.5.3. *Let m_i and m_k be any two men on a woman w_j's list. Then*

1. *$m_i \lhd_{w_j} m_k$ if either*

   (i) $(m_i, w_j) \in U$, $(m_k, w_j) \in U$, m_i is promoted and m_k is unpromoted or

   (ii) $(m_i, w_j) \in A$, $(m_k, w_j) \in U$ and m_k is unpromoted.

2. *$m_i \lhd'_{w_j} m_k$ if $m_i \not\trianglelefteq_{w_j} m_k$, $m_k \not\trianglelefteq_{w_j} m_i$ and w_j prefers m_i to m_k in the classical sense.*

We define $\prec_{w_j} = \lhd_{w_j} \cup \lhd'_{w_j}$.

The relation $\prec_{w_j}$ will be used to determine whether a proposal from a man is accepted or rejected by w_j.

The main algorithm `Approx-smiss` (as shown in Algorithm 4.1) starts by calling `Mod-exgs` (as shown in Algorithm 4.2) where a *proposal sequence* is started by allowing each man to propose to women beginning from the first woman on his preference list. If a man m_i proposes to a woman w_j on his list and w_j is matched and $m_i \prec_{w_j} M(w_j)$, then w_j is unmatched from her partner m_k and m_k will be allowed to continue proposing to other women on his list. w_j is then assigned to m_i. On the other hand, if $M(w_j) \prec_{w_j} m_i$ then w_j rejects m_i's proposal. Also if w_j is unmatched when m_i proposes, she is assigned to m_i. Irrespective of whether the proposal from m_i is accepted or rejected, if $(m_i, w_j) \in A$ then all pairs (m_k, w_j) such that $rank(w_j, m_k) > rank(w_j, m_i)$ are deleted from the instance. However if $(m_i, w_j) \in U$ no such deletions take place. This proposal sequence continues until every man is either matched or has exhausted his preference list.

After each proposal sequence (where control is returned to the `Approx-smiss` algorithm), if a promoted man still remains unmatched after proposing to all the women on his preference list, he is removed from the instance. Also if a previously unpromoted man exhausts his preference lists and is still unmatched, he is promoted and a new proposal sequence is initiated (by calling `Mod-exgs`). The algorithm terminates when each man either (i) is assigned a partner, (ii) has no woman on his preference list or (iii) has been promoted and has proposed to all the women on his preference list for a second time.

Lemma 4.5.4. *If Algorithm* `Approx-smiss` *is executed on an* SMISS *instance (I, G), it terminates with a socially stable matching M in (I, G).*

Algorithm 4.1 `Approx-smiss`

initial matching $M = \emptyset$;
while some unmatched man with a non-empty preference list exists **do**
  call `Mod-exgs`;
  for all m_i such that m_i is unmatched and promoted **do**
    remove m_i from instance;
  for all m_i such that m_i is unmatched, unpromoted and has a non-empty preference list **do**
    promote m_i;
return the resulting matching M;

Algorithm 4.2 `Mod-exgs`

while some man m_i is unmatched and still has a woman left on his list **do**
  $w_j = next(m_i)$;
  if w_j is matched in M and $m_i \prec_{w_j} M(w_j)$ **then**
    $M = M \setminus \{(M(w_j), w_j)\}$;
  if w_j is unmatched in M **then**
    $M = M \cup \{(m_i, w_j)\}$;
  if $(m_i, w_j) \in A$ **then**
    for each m_k such that $(m_k, w_j) \in \mathcal{A}$ and $rank(w_j, m_k) > rank(w_j, m_i)$ **do**
      delete (m_k, w_j) from instance;

Proof. Suppose M is not a socially stable matching and some pair (m_i, w_j) socially blocks M in (I, G). Hence $(m_i, w_j) \in A$. If w_j is unmatched in M then she never received a proposal from m_i (as if she did, she will never become unmatched afterwards). This implies that m_i must prefer his partner in M to w_j as he never proposed to w_j. Thus (m_i, w_j) cannot socially block M in this case.

On the other hand, suppose w_j is matched in M but prefers m_i to $M(w_j) = m_k$. Also suppose m_i is either unmatched in M or prefers w_j to $M(m_i)$. Then m_i proposed to w_j during the algorithm's execution or (m_i, w_j) was deleted. In either case, all successors of m_i on w_j's list will be deleted, so $(m_k, w_j) \notin M$, a contradiction. □

Lemma 4.5.5. *During any execution of the algorithm* `Mod-exgs`*, if m_i proposes to w_j, $rank(w_j, m_i) < rank(w_j, M(w_j))$ and $(m_i, w_j) \in A$ then w_j will never reject m_i.*

Proof. This follows from our definition of the $\prec_{w_j}$ relation. Suppose that w_j rejects m_i for a some man m_k and $rank(w_j, m_i) < rank(w_j, m_k)$. Thus $m_k \prec_{w_j} m_i$. This implies that $m_k \lhd_w m_i$ or $m_k \lhd'_w m_i$. Since $(m_i, w_j) \in A$ then $m_k \not\lhd_w m_i$ so $m_k \lhd'_w m_i$ which in turn implies that $rank(w_j, m_k) < rank(w_j, m_i)$, a contradiction to our assumption. □

The execution of the `Mod-exgs` algorithm takes $O(m)$ time where $m = |\mathcal{A}|$ is the number of acceptable pairs. These executions can be performed at most $2n_1$ times, where n_1 is the number of men, as a man is given at most two chances to propose to

the women on his list. Thus the overall time complexity of the algorithm is $O(n_1 m)$. The above results, together with Theorem 4.5.1, lead us to state the following theorem concerning the performance guarantee of `Approx-smiss`.

Theorem 4.5.6. MAX HRSS *is approximable within a factor of 3/2.*

Proof. We prove this result by adopting techniques similar to those used by Iwama et al. [67] and subsequently by Király [75]. Let M be the matching obtained from running `Approx-smiss` on an SMISS instance (I, G). From 4.5.4 we know that M is socially stable. We consider alternating paths of odd-length in connected components of the union $M \cup M_{opt}$. An alternating path is one in which the edges belong alternatively to M and M_{opt}. An odd-length alternating path P with more edges in M_{opt} than edges in M can be used to increase the size of M by removing $P \cap M$ from M and adding $P \cap M_{opt}$ to M. It is easy to see that, for alternating paths of length greater than 3, the number of edges in M_{opt} is at most 3/2 times the number of edges in M. We now show that alternating paths of length 3 cannot exist in $M \cup M_{opt}$.

Consider an alternating path of length 3 $\langle (m, w'), (m, w), (m', w) \rangle$ such that $(m, w') \in M_{opt}$, $(m, w) \in M$ and $(m', w) \in M_{opt}$. We will show that (m, w) socially blocks M_{opt} thus forming a contradiction. Since w' is unmatched in M, she was never proposed to during the entire execution of `Approx-smiss`. So w' did not delete any men from her preference list and m is unpromoted (in order to be promoted, he would first have had to proposed to w'). Thus m prefers w to w'. Also since m' is unmatched in M, either (i) the pair (m', w) was deleted from the instance at some point during the execution of `Approx-smiss` or (ii) w rejected m' twice during the execution of `Approx-smiss`.

Consider case (i): if (m', w) was deleted during the execution of the algorithm then w received a proposal from some man m'' such that $(m'', w) \in A$ and w prefers m'' to m'. Thus all successors of m'' on w's list would also have been deleted and so w prefers m to m'' and hence to m'. Although w would have accepted the proposal from m'' temporarily (see Lemma 4.5.5), she ended up with an unpromoted man m. It might be that $m = m''$. On the other hand, suppose she accepted a series of proposals from men after m'' proposed to w $\langle m''_0, m''_1, ..., m''_k \rangle$ for some $k \geq 0$ before finally being assigned to m. Thus $m''_0 \prec_w m''$, which implies that $m''_0 \lhd_w m''$ or $m''_0 \lhd'_w m''$. Since $(m'', w) \in A$, $m''_0 \not\lhd_w m''$ which means $m''_0 \lhd'_w m''$. But for that to be true, $m'' \not\lhd_w m''_0$ as well. For that to happen, m''_0 must not be an unacquainted unpromoted man on w's list. This means m''_0 can be an acquainted man or an unacquainted promoted man on w's list. If $m = m''_0$ (i.e. $k = 0$) then we know that m is unpromoted and so $(m, w) \in A$. On the other hand if another man m''_1 exists in the sequence, then the same argument follows that $m''_1 \prec_w m''_0$ means that $m''_1 \lhd_w m''_0$ or $m''_1 \lhd'_w m''_0$. As already observed, m''_0 is either an acquainted man or an unacquainted, promoted man on w's list. In both cases,

$m_1'' \not\trianglelefteq_w m_0''$ meaning $m_1'' \trianglelefteq'_w m_0''$. This implies that $m_0'' \not\trianglelefteq_w m_1''$. But for that condition to be satisfied, m_1'' must not be an unacquainted unpromoted man on w's list. Once again this means m_1'' can be an acquainted man or an unacquainted promoted man on w's list. The same sequence can continue for all men in the sequence until m proposes. Since we already established that m is unpromoted, it follows that $(m, w) \in A$.

Now consider case (ii): w rejected m' even when he was promoted because of a proposal from some man m''. Thus $m'' \prec_w m'$ means $m'' \trianglelefteq_w m'$ or $m'' \trianglelefteq'_w m'$. Since we know that m' was promoted, then $m'' \not\trianglelefteq_w m'$ thus $m'' \trianglelefteq'_w m'$. This means that $m' \not\trianglelefteq_w m''$ and w prefers m'' to m'. Since $m' \not\trianglelefteq_w m''$, m'' must not be an unacquainted unpromoted man on w's list as m' is promoted. Thus m'' must be promoted or an acquainted man on w's list. If m'' were an acquainted man on w's list, the pair (m', w) would be deleted and the logic presented in case (i) above would follow through. Now suppose m'' is promoted and unacquainted on w's list. We know that $m \neq m''$ as m was unpromoted in M. Then w may have accepted a series of proposals from men $\langle m_0'', m_1'', ..., m_k'' \rangle$ for some $k \geq 0$ before finally being assigned to m. Thus $m_0'' \prec_w m''$ means $m_0'' \trianglelefteq_w m''$ or $m_0'' \trianglelefteq'_w m''$. Since m'' is promoted and unacquainted on w's list $m_0'' \not\trianglelefteq_w m''$ meaning $m_0'' \trianglelefteq'_w m''$. This implies that $m'' \not\trianglelefteq_w m_0''$. But for that condition to be satisfied, m_0'' must not be an unacquainted unpromoted man on w's list. A similar argument to the one presented for case (i) above results in the conclusion that $(m, w) \in A$.

The following conditions (i) m prefers w to w' (ii) w prefers m to m' and (iii) $(m, w) \in A$ imply that (m, w) will socially block M_{opt}, a contradiction. □

The SMISS instance shown in Figure 4.2 shows that the 3/2 bound for the algorithm is tight. Here $M_{opt} = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$ is the unique maximum socially stable matching. Also the approximation algorithm outputs $M = \{(m_1, w_1), (m_2, w_2)\}$ irrespective of the order in which proposals are made. Clearly this instance can be replicated to obtain an arbitrarily large SMISS instance for which the performance guarantee is tight.

We remark that a similar 3/2-approximation algorithm for MAX HRSS was presented independently by Askalidis et al. in [13].

4.5.2 Inapproximability result

We now show lower bounds on the approximability of our problem. We start by giving the inapproximability result assuming P≠NP.

Theorem 4.5.7. MAX SMISS *is not approximable within* $21/19 - \varepsilon$ *for any* $\varepsilon > 0$, *unless P=NP.*

men's preferences	women's preferences
$m_1 : w_1 \quad w_3$	$w_1 : m_2 \quad m_1$
$m_2 : w_1 \quad w_2$	$w_2 : m_2 \quad m_3$
$m_3 : w_2$	$w_3 : m_1$

social network graph G

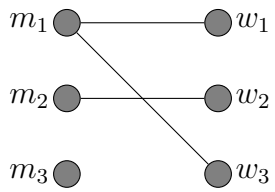

Figure 4.2: $|M_{opt}| = (3/2).|M|$

Proof. We rely on a proof of the NP-hardness of approximating MAX SMTI given in [40]. It is shown, by a reduction from VERTEX COVER, that there is no approximation algorithm for MAX SMTI with performance guarantee of $21/19 - \varepsilon$ for any $\varepsilon > 0$, unless P=NP. This is shown to be true even for instances where ties appear on one side only, each preference list is either strictly ordered or has a single tie of length 2 and ties appear at the end of the preference lists. Thus the same reduction shown in the proof of Theorem 4.4.1 can be used to construct an SMISS instance (I, G) such that a polynomial-time algorithm that approximates MAX SMISS to within a factor of $21/19 - \varepsilon$ would do the same for MAX SMTI. □

A better lower bound on the approximability of MAX SMISS of $3/2 - \varepsilon$ for any constant $\varepsilon > 0$ assuming the *Unique Games Conjecture*, was presented by [12].

4.6 Some special cases of HRSS

Given the hardness results obtained for the problem of finding a maximum socially stable matching in a general HRSS instance, the need arises to investigate special cases of the problem that are tractable. This section describes some polynomial-time solvability results for three special cases of HRSS. Subsection 4.6.1 gives a polynomial-time algorithm for finding a maximum socially stable matching given an instance of $(2, \infty)$-MAX SMISS. In Subsection 4.6.2 we provide a polynomial-time algorithm for MAX HRSS in the case where there is a constant number of unacquainted pairs and in Subsection 4.6.3 we consider the case where the number of acquainted pairs is constant, again providing a polynomial time-algorithm for MAX HRSS in that context.

4.6.1 $(2, \infty)$-MAX SMISS

Given an SMISS instance (I, G), where each man is allowed to have at most two women in his preference list and each woman is allowed to have an unbounded-length preference list, we show that a maximum socially stable matching can be found in polynomial time. We make slight modifications to the algorithm used to find a maximum stable matching in an instance of $(2, \infty)$-MAX SMTI (the problem of finding a maximum stable matching given an SMTI instance where each man has at most two women on his preference list) described in [61]. The resulting algorithm, which we call $(2, \infty)$-`Max-smiss-alg`, is broken down into three phases.

In Phase 1, some pairs that cannot be involved in any socially stable matching in (I, G) are *deleted* from the instance. A pair (m_i, w_j) is deleted by removing m_i from w_j's preference list and vice versa. We call the resulting preference lists the *reduced preference lists*. For each man m_i, if the first woman w_j on m_i's preference list satisfies $(m_i, w_j) \in A$, where $A = E(G)$ is the set of acquainted pairs in (I, G), we delete all pairs (m_k, w_j) for all successors m_k of m_i on w_j's preference list. Pair (m_k, w_j) cannot be involved in any socially stable matching as (m_i, w_j) will socially block any matching they were involved in.

In Phase 2, we construct a weighted bipartite graph G' from the resulting instance with a reduced preference list. This is done by representing the men and women as nodes on the two sides of the graph and adding an edge between a man m_i on one side and a woman w_j on the other if w_j appears on m_i's preference list. The weight placed on the edge will be the position of m_i on w_j's preference list (denoted by $rank(w_j, m_i)$). Algorithm 4.4 describes the process. A minimum cost maximum cardinality matching $M_{G'}$ in the the resulting bipartite graph is then generated using the algorithm described in [28].

At this stage it is not guaranteed that the resulting maximum matching $M_{G'}$ is socially stable. $M_{G'}$ may admit a social blocking pair (m_i, w_j), where $(m_i, w_j) \in A$, w_j is the first-choice partner of m_i, m_i is assigned to his second-choice partner w_k in $M_{G'}$ and w_j is unassigned in $M_{G'}$ (as we will show later, this the only form of social blocking pair that $M_{G'}$ can admit). To remove such blocking pairs, during Phase 3, we assign m_i to w_j thus making w_k unmatched. At this stage w_k may herself be the first-choice partner of some man m_l where $(m_l, w_k) \in A$, so a similar operation is performed involving m_l and w_k. The process continues until there is no man who is matched to his second-choice woman while forming an edge in G with his unmatched first-choice woman. Algorithm 5.1 shows the entire $(2, \infty)$-`Max-smiss-alg` algorithm.

We now show that at the end of this phase, for $(2, \infty)$-MAX SMISS instances, the matching produced is both socially stable and of maximum size.

Algorithm 4.3 $(2, \infty)$-`Max-smiss-alg`

```
1: /* Phase 1 */
2: while some man m_i has a first-choice woman w_j such that (m_i, w_j) ∈ A do
3:    for each successor m_k of m_i on w_j's list do
4:       delete the pair (m_k, w_j);
5: /* Phase 2 */
6: G' := BuildGraph();
7: M_G' := minimum weight maximum matching in G';
8: /* Phase 3 */
9: M := M_G';
10: while there exists a man m_i who is matched to his second-choice woman w_k
    and his first-choice w_j is an unmatched woman such that (m_i, w_j) ∈ A do
11:    M := M\{(m_i, w_k)};
12:    M := M ∪ {(m_i, w_j)};
13: return M;
```

Algorithm 4.4 `BuildGraph`

```
1: V := U ∪ W; /* U and W are sets of men and women in I */
2: E' := ∅;
3: for each man m_i ∈ U do
4:    for each woman w_j on m_i's reduced list do
5:       E' := E' ∪ {(m_i, w_j)};
6:       weight(m_i, w_j) := rank(w_j, m_i);
7: G' := (V, E');
8: return G';
```

Lemma 4.6.1. $(2, \infty)$-`Max-smiss-alg` *terminates.*

Proof. It is easy to see that Phases 1 and 2 terminate. For every iteration of Phase 3, one man always improves from his second to his first choice while no man obtains a worst partner or becomes unmatched. Since the total number of possible improvements is finite, it is clear the phase is bound to terminate. □

Lemma 4.6.2. *Phase 1 of* $(2, \infty)$-`Max-smiss-alg` *never deletes a socially stable pair, which is a man-woman pair that belongs to some socially stable matching in* (I, G).

Proof. Suppose pair (m_i, w_j) is deleted during some execution of $(2, \infty)$-`Max-smiss-alg` and $(m_i, w_j) \in M$ where M is some socially stable matching in (I, G). Then m_i was deleted from w_j's preference list because w_j was the first-choice woman of some man m_k such that $(m_k, w_j) \in A$ and w_j prefers m_k to m_i. Therefore (m_k, w_j) will socially block M, a contradiction. □

Lemma 4.6.3. *The matching returned by* $(2, \infty)$-`Max-smiss-alg` *is socially stable in* (I, G).

Proof. Suppose the matching M produced by $(2,\infty)$-`Max-smiss-alg` is not socially stable in (I,G). Then some pair $(m_i,w_j) \in A$ socially blocks M in I. For this to occur, one of the following cases must arise.

Case (i): m_i and w_j are unmatched in M. Then m_i is unmatched in $M_{G'}$ and either w_j was initially unmatched in $M_{G'}$ or became unmatched due to some operation in Phase 3 of $(2,\infty)$-`Max-smiss-alg`. If w_j was initially unmatched in $M_{G'}$ then $M_{G'}$ could have been increased in size by adding (m_i,w_j) thus contradicting the fact that $M_{G'}$ is of maximum cardinality. Suppose w_j became unmatched due to Phase 3. Let m_{p_1} denote w_j's partner in $M_{G'}$. During Phase 3, m_{p_1} must have become assigned to his first-choice woman w_{q_1}. Suppose w_{q_1} was unmatched in $M_{G'}$. Then a larger matching can be obtained by augmenting $M_{G'}$ along the path $(m_i,w_j),(w_j,m_{p_1}),(m_{p_1},w_{q_1})$ contradicting to the fact that M_G is of maximum cardinality. Thus w_{q_1} must have been assigned in $M_{G'}$ and became unmatched during Phase 3 as well. If w_{q_1} was assigned to m_{p_2} in $M_{G'}$, m_{p_2} must have become assigned to his first-choice woman w_{q_2} during Phase 3. Using a similar argument to that used for w_{q_1}, we can argue that w_{q_2} must have been assigned in $M_{G'}$ as well. Thus some man moved from w_{q_2} to his first-choice woman. This process may be continued and at each iteration of Phase 3, some man must strictly improve and no man becomes worse off. Since the possible number of such improvements is finite, there are a finite number of women that can be unmatched in this way in Phase 3. Hence at some point, there exists a man m_{p_s}, who switches to his first-choice woman w_{q_s} and w_{q_s} was already unmatched in $M_{G'}$. We can then construct an augmenting path in G' of the form $(m_i,w_j),(w_j,m_{p_1}),(m_{p_1},w_{q_1}),(w_{q_1},m_{p_2}),(m_{p_2},w_{q_2}),...,(m_{p_s},w_{q_s})$ which contradicts the fact that $M_{G'}$ is of maximum cardinality.

Case (ii): m_i is unmatched in M and w_j prefers m_i to $m_l = M(w_j)$. Then m_i is unmatched in $M_{G'}$ as well. Suppose that w_j was assigned to m_l in $M_{G'}$. Since w_j prefers m_i to m_l, a matching of equal size to $M_{G'}$ but with a lower weight can be obtained by pairing m_i with w_j, leaving m_l unmatched, thus contradicting the fact that $M_{G'}$ is a minimum weight maximum cardinality matching. Thus w_j is not assigned to m_l in $M_{G'}$. Then w_j is either unmatched in $M_{G'}$ or is assigned in $M_{G'}$ to some man m_p, where $m_p \neq m_l$ and $m_p \neq m_i$. If w_j is unmatched in $M_{G'}$, the fact that $M_{G'}$ is a maximum cardinality matching is contradicted. If w_j is assigned to m_p in $M_{G'}$ then since w_j is no longer assigned to m_p in M, m_p must have switched to his first-choice woman w_q during Phase 3. Hence either w_q was unmatched in $M_{G'}$ or w_q became unmatched due to some man being switched to his first-choice woman. Using a similar argument as in case (i), we can construct an augmenting path that contradicts the fact that $M_{G'}$ is of maximum cardinality.

Case (iii): m_i is assigned to w_k in M and m_i prefers w_j to w_k and w_j is unmatched in M. Since m_i's preference list is of length 2, w_j is m_i's first-choice and $(m_i,w_j) \in A$

and so this case satisfies the loop condition in Phase 3 and thus should never arise once Phase 3 has terminated (as it must, by Lemma 4.6.1).

Case (iv): m_i is assigned to w_k in M and m_i prefers w_j to w_k and w_j is assigned to m_l in M and w_j prefers m_i to m_l. Once again, since m_i's list is of length 2, then w_j must be m_i's first-choice and $(m_i, w_j) \in A$. Therefore the loop condition of Phase 1 would have ensured that man m_l was deleted from w_j's preference list. □

Since, by Lemma 4.6.2, Phase 1 never deletes a socially stable pair, a maximum socially stable matching must consist of pairs that belong to the reduced lists. Since $M_{G'}$ is a maximum matching and Phase 3 never reduces the size of the matching, it follows that the matching produced by the algorithm is of maximum cardinality. Finally since by Lemma 4.6.3, the matching produced is a socially stable matching, it follows that the algorithm produces a maximum socially stable matching in (I, G).

The complexity of the algorithm is dominated by Phase 2. A minimum cost maximum matching in $G' = (V, E')$ can be found in $O(\sqrt{|V|}|E'|\log|V|)$ [28]. Let $n = |V| = n_1 + n_2$ be the total number of men and women. Since the set $\mathcal{A}$ of acceptable pairs satisfies $|\mathcal{A}| \leq 2n_1 = O(n)$, it follows that $(2, \infty)$-`Max-smiss-alg` has a time complexity of $O(n^{3/2} \log n)$. We have thus proved the following theorem.

Theorem 4.6.4. *Given an instance (I, G) of $(2, \infty)$-MAX SMISS, Algorithm $(2, \infty)$-*`Max-smiss-alg` *generates a maximum socially stable matching in $O(n^{3/2} \log n)$ time, where n is the total number of residents and hospitals in (I, G).*

4.6.2 HRSS with a constant number of unacquainted pairs

It is easy to see that in the special case where the set U of unacquainted pairs is exactly the set $\mathcal{A}$ of acceptable pairs in the underlying HR instance, then the set A of acquainted pairs satisfies $A = \emptyset$ and every matching found is a socially stable matching. Also if the instance contains no unacquainted pairs (i.e., $A = \mathcal{A}$ and $U = \emptyset$), then only stable matchings in the classical sense are socially stable. In both these cases, a maximum socially stable matching can be generated in polynomial time. The case may however arise where the number of unacquainted pairs is constant. In this case, we show that it is also possible to generate a maximum socially stable matching in polynomial time.

Let (I, G) be an instance of HRSS and let $S \subseteq \mathcal{A}$ be a subset of the acceptable pairs in I. We denote $I \backslash S$ as the instance of HR obtained from I by deleting the pairs in S from the preference lists in I. The following proposition plays a key role in establishing the correctness of the algorithm.

Lemma 4.6.5. *Let (I, G) be an instance of* HRSS. *Let M be a socially stable matching in (I, G). Then there exists a set of unacquainted pairs $U' \subseteq U$ such that M is stable in $I' = I \backslash U'$. Conversely suppose that M is a stable matching in $I' = I \backslash U'$ for some $U' \subseteq U$. Then M is socially stable in (I, G).*

Proof. Suppose M is socially stable in (I, G). Let $U' = U \backslash (M \cap U) = U \backslash M$. We claim that M is stable in $I' = I \backslash U'$. Suppose (r_i, h_j) blocks M in I'. Then $(r_i, h_j) \notin M$ and edges in I' are those in $A \cup M \cap U$ and $(r_i, h_j) \notin M \cap U$. Thus $(r_i, h_j) \in A$ making M socially unstable in (I, G), a contradiction.

Conversely, suppose that M is stable in $I' = I \backslash U'$ for some $U' \subseteq U$. We claim that M is socially stable in (I, G). Suppose that (r_i, h_j) socially blocks M in (I, G), then $(r_i, h_j) \in A$ so $(r_i, h_j) \notin U$. Thus (r_i, h_j) is a pair in I' and so (r_i, h_j) blocks M in I', a contradiction. □

By considering all subsets $U' \subseteq U$, forming I', finding a stable matching in each such I' and keeping a record of the maximum stable matching found, we obtain a maximum socially stable matching in (I, G). This discussion leads to the following theorem.

Theorem 4.6.6. *Given an instance (I, G) of* HRSS, *a maximum socially stable matching can be generated in $O(2^k|\mathcal{A}|)$ time, where $|\mathcal{A}|$ is the number of acceptable pairs and $k = |U|$ is the number of unacquainted pairs.*

Proof. By considering all subsets $U' \subseteq U$, forming I', finding stable a matching in each such I' and keeping a record of the maximum stable matching found, we obtain a maximum socially stable matching in (I, G). Let M be the largest stable matching found over all $I' = I \backslash U'$ where $U' \subseteq U$. Then by Lemma 4.6.5, M is socially stable in (I, G). Suppose for a contradiction that M' is a socially stable matching in I where $|M'| > |M|$. Then by Lemma 4.6.5 there exists some $U'' \subseteq U$ such that M' is stable in I''. But that contradicts M as the largest stable matching in $I \backslash U'$ taken over all $U' \subseteq U$.

Finding all subsets of U can be done in $O(2^k)$ time where $k = |U|$. Forming I' and finding a stable matching in I' can be done in $O(|\mathcal{A}|)$ time where $\mathcal{A}$ is the set of acceptable pairs in I. Hence the overall time complexity of the algorithm is $O(2^k|\mathcal{A}|)$. □

4.6.3 HRSS with a constant number of acquainted pairs

We now consider the restriction of HRSS in which the set A of acquainted pairs is small. In this section we present an algorithm for finding a maximum socially stable matching

given an instance (I, G) of HRSS with this restriction. Let $A = \{e_1, e_2, ..., e_k\}$ where $k = |A|$ and e_i represents an acquainted pair (r_{s_i}, h_{t_i}) $(1 \leq i \leq k)$. A tree T of depth k is constructed with all nodes at depth i labelled e_{i+1} $(i \geq 0)$. There are left and right branches below e_i. Each branch corresponds to a condition placed on r_{s_i} or h_{t_i} with respect to a matching M. The left branch below e_i (i.e., a resident condition branch) corresponds to the condition that r_{s_i} is matched in M and prefers his partner to h_{t_i}. The right branch below e_i (i.e., a hospital condition branch) corresponds to the condition that h_{t_i} is fully subscribed in M and has no partner worse than r_{s_i}. Satisfying at least one of these conditions ensures that M admits no blocking pair involving (r_{s_i}, h_{t_i}). The tree is constructed in this manner with the nodes at depth $k-1$, labelled e_k, branching in the same way to dummy leaf nodes e_{k+1} (not representing acquainted pairs).

A path P from the root node e_1 to a leaf node e_{k+1} will visit all pairs in A exactly once. Every left branch in P gives a resident condition and every right branch gives a hospital condition. Let R' and H' be the set of residents and hospitals involved in resident and hospital conditions in P respectively. Given a matching M, enforcing all the conditions along P can be achieved by first deleting all pairs from the instance I that could potentially violate these conditions. So if some resident condition along P states that a resident r_{s_i} must be matched in M to a hospital he prefers to h_{t_i} then r_{s_i}'s preference list is truncated starting with h_{t_i}. If some hospital condition states that a hospital h_{t_i} must be fully subscribed in M and must not be matched to a resident worse than r_{s_i} then h_{t_i}'s preference list is truncated starting from the resident immediately following r_{s_i}. After performing these truncations based on the conditions along P, a new HR instance I' is obtained.

Lemma 4.6.7. *If M is a matching in I' that is computed at the leaf node of a path P and all residents in R' are matched in M and all hospitals in H' are fully subscribed in M then M is a socially stable matching in (I, G).*

Proof. Suppose some resident-hospital pair socially blocks M in (I, G). Then this pair belongs to A so it corresponds to a node $e_i = (r_{s_i}, h_{t_i})$ in T for some i $(1 \leq i \leq k)$. So in M, either (i) r_{s_i} is unmatched or prefers h_{t_i} to $M(r_{s_i})$ and (ii) either h_{t_i} is undersubscribed or prefers r_{s_i} to $M(h_{t_i})$. Suppose in T, the left hand branch from e_i was chosen when following a path P from the root to a leaf node. Then $r_{s_i} \in R'$, r_{s_i} is matched and by the truncations carried out, r_{s_i} has a better partner than h_{t_i}, a contradiction. Thus the right hand branch from e_i was chosen when following P. Then $h_{t_i} \in H'$, h_{t_i} is fully subscribed and by the truncations h_{t_i} has no partner worse than r_{s_i}. But $(r_{s_i}, h_{t_i}) \notin M$, so h_{t_i} has a partner better than r_{s_i}, a contradiction. □

With I' obtained due the truncations carried out by satisfying conditions along a path P from the root node to a leaf node, we then seek to obtain a matching in which all the

residents in R' are matched and hospitals in H' are fully subscribed. For each hospital h_j in I', we define a set of clones of h_j, $\{h_{j,1}, h_{j,2}, ..., h_{j,c_j}\}$, corresponding to the number of posts c_j available in h_j. Let $H'' = \{h_{j,k} : h_j \in H' \wedge 1 \leq k \leq c_j\}$ denote the set of all clones obtained from hospitals in H'. We define a bipartite graph G' where one set of nodes is represented by the set of residents in I' and the other set of nodes is represented by the hospital clones in I'. If r_i finds h_j acceptable in I, we add an edge from r_i to $h_{j,q}$ in G' for all q $(1 \leq q \leq c_j)$. We define a new graph G'' containing the same nodes and edges in G' with weights placed on the edges. We mark all the nodes representing the residents in R' and the hospital clones in H'' as red nodes with the remaining nodes uncoloured. We place weights on the edges as follows: (i) an edge between a red node and an uncoloured node is given a weight of 1; (ii) an edge between two red nodes is given weight of 2; (iii) an edge between two uncoloured nodes is given a weight of 0. We then find a maximum weight matching M' in the resulting weighted bipartite graph G''. Let $wt(M')$ denote the weight of a matching M' in G''. Then

$$wt(M') = |\{(r_i, h_{j,q}) \in M' : r_i \in R'\}| + |\{(r_i, h_{j,q}) \in M' : h_{j,q} \in H''\}| \leq |R'| + |H''|.$$

Moreover $wt(M') = |R'| + |H''|$ if and only if every agent in $R' \cup H''$ is matched in M'. For such a matching M' in G'', by construction, M' is also a matching in G' and a maximum cardinality matching M'' can be obtained in G' by continuously augmenting M' until no augmenting path can be found. Since any node already matched in M' will remain matched in M'' it follows that all the residents and hospital clones in $R' \cup H''$ will be matched in M'' and such a matching, by Lemma 4.6.7, will be socially stable in (I, G). If however, $wt(M') < |R'| + |H''|$ then some resident or hospital clone in $R' \cup H''$ remains unmatched in any maximum matching in G' introducing the possibility of a social blocking pair of M' in (I, G). In this case, P is ruled as infeasible and another path is considered, otherwise P is called feasible.

There are 2^k paths from the root node to leaf nodes in the tree T. The following proposition is important to our result.

Lemma 4.6.8. *There must exist at least one feasible path in T.*

Proof. Let M be a stable matching in I. Then M is socially stable in (I, G). Consider each node $e_i = (r_{s_i}, h_{t_i})$ in T. If $(r_{s_i}, h_{t_i}) \in M$, we branch right at e_i. If $(r_{s_i}, h_{t_i}) \notin M$, and r_{s_i} is matched and has a partner better than h_{t_i}, we branch left at e_i. If $(r_{s_i}, h_{t_i}) \notin M$, and h_{t_i} is fully subscribed and has no partner worse than r_{s_i}, we branch right at e_i. Any other condition would mean that (r_{s_i}, h_{t_i}) would block M in I, a contradiction. This process, starting from the root node e_1, gives a feasible path P through T to some leaf node v where, for the set of residents R' and hospitals H' involved in P, $wt(M) = |R'| + |H'|$. □

To generate a maximum socially stable matching M in an instance (I, G) of HRSS, we first compute the matchings at the leaf nodes of each of the 2^k paths through T from the root node to leaf nodes. Next, of all the matchings obtained from feasible paths, we find the maximum matching M''. A maximum socially stable matching M can then be constructed by letting

$$M = \{(r_i, h_j) : (r_i, h_{j,k}) \in M'' \text{ for some } k \ (1 \leq k \leq c_j)\}.$$

Lemma 4.6.9. *If M is a matching obtained from the process described above, M is a maximum socially stable matching in (I, G).*

Proof. Lemma 4.6.7 shows that M is socially stable in (I, G). Suppose M' is a socially stable matching in (I, G) such that $|M'| > |M|$. Construct a feasible path P through T from the root node to a leaf node v, branching left or right at each node $e_i = (r_{s_i}, h_{t_i})$ as follows. If r_{s_i} is matched in M' to some hospital better than h_{t_i}, branch left. Otherwise as (r_{s_i}, h_{t_i}) is not a social blocking pair of M', h_{t_i} is fully subscribed in M' with no assignee worse than r_{s_i}, in which case branch right. As before, construct sets R' and H' as follows. For every left branch in P involving a resident r_{s_i}, add r_{s_i} to R' and for every right branch in P involving a hospital h_{t_i}, add h_{t_i} to H'. Matching M' then satisfies the property that every resident $r_{s_i} \in R'$ is matched in M' to a hospital better than h_{t_i} and every hospital $h_{t_i} \in H'$ is fully subscribed with residents no worse than r_{s_i}. At the leaf node v of P, the algorithm constructs a matching M'' which is of maximum cardinality with respect to the restrictions that every resident $r_{s_i} \in R'$ is matched to a hospital better than h_{t_i} and every hospital $h_{t_i} \in H'$ is fully subscribed with residents no worse than r_{s_i}. Hence $|M''| \geq |M'|$ and since $|M'| > |M|$, it follows that M'' contradicts the choice of M as the largest matching taken over all leaf nodes. □

The above proposition leads to the following main result of this subsection.

Theorem 4.6.10. *Given an instance (I, G) of HRSS where the set A of acquainted pairs satisfies $|A| = k$, a maximum socially stable matching can be generated in $O(2^k c_{max} m \sqrt{n_1 + C})$ time where n_1 is the number of residents, m is the number of acceptable pairs, c_{max} is the largest capacity of any hospital and C is the total capacity of all the hospitals in the problem instance.*

Proof. The number of paths from the root node to a leaf node is 2^k. Performing the truncations imposed by the conditions along a path can be done in $O(m)$ time where m is the number of acceptable pairs in I. The number of nodes n' and the number of

edges m' in G' are given by:

$$m' = \sum_{j=1}^{n_2} |pref(h_j)| c_j \leq c_{max} \sum_{j=1}^{n_2} |pref(h_j)| \leq c_{max} m$$
$$n' = n_1 + \sum_{j=1}^{n_2} c_j = n_1 + C$$

where n_1 is the number of residents, n_2 is the number of hospitals, $pref(h_j)$ is the set of residents in h_j's preference list, c_{max} is the largest capacity of any hospital and C is the total number of posts in the problem instance. Finding a maximum weight matching in G'' can be done in $O(m'\sqrt{n'})$ time [24] (since the edge weights have $O(1)$ size). Augmenting such a matching to a maximum cardinality matching in G' can be done in $O(m'\sqrt{n'})$ time [46]. Thus the time complexity of the algorithm is $O(2^k c_{max} m\sqrt{n_1 + C})$. □

We conclude this section with the following results, which are an immediate consequence of Theorems 4.6.6 and 4.6.10.

Corollary 4.6.11. MAX HRSS *is in FPT with parameter k where k is the number of unacquainted pairs.*

Corollary 4.6.12. MAX HRSS *is in FPT with parameter k where k is the number of acquainted pairs.*

4.7 Empirical evaluation

4.7.1 Introduction

The `Approx-smiss` algorithm was implemented and evaluated in Java. The objective was to measure how the mean execution times and matching sizes change as we vary the size and the social network density of randomly generated SMISS instances. An IP model for MAX SMISS was also encoded and solved using the CPLEX 12.5.1 IP solver. The social stability of the matchings produced by both implementations was separately verified by a subroutine specifically implemented to check for this property.

The IP model for MAX HRSS presented in Figure 4.3, is a slight modification to the one presented for MAX HRT in Chapter 3. The only change/alteration to the model is that, in the SMISS context we ensure that the stability-enforcing constraints are applied only to acquainted pairs (see Constraint 3 of Figure 4.3). We present the entire IP model below for completeness. The proof of correctness presented in [79] carries over to the HRSS case.

$$\max \sum_{i=1}^{n_1} \sum_{h_j \in P(r_i)} x_{i,j}$$

subject to

1. $\sum_{h_j \in P(r_i)} x_{i,j} \leq 1 \qquad \forall \ (1 \leq i \leq n_1)$
2. $\sum_{r_i \in P(h_j)} x_{i,j} \leq c_j \qquad \forall \ (1 \leq j \leq n_2)$
3. $c_j \left(1 - \sum_{h_q \in S_{i,j}} x_{i,q}\right) - \sum_{r_p \in T_{i,j}} x_{p,j} \leq 0 \quad \forall \ (r_i, h_j) \in G$

$x_{i,j} \in \{0, 1\}$

Figure 4.3: A MAX HRSS IP model

Random SMISS instances were generated using an instance generator and solved using both the `Approx-smiss` algorithm and the IP solver. Results from the implemented algorithm were compared with those produced by the IP solver. This improved our confidence in the correctness of both implementations. The instance generator can be configured to vary certain properties of the instances produced. The results presented in this section involve varying:

1. The number of men n_1 and women n_2 in the underlying SMI instance (always keeping $n_1 = n_2$) and
2. The *density* d_G of the social network, which is the probability that a randomly selected acceptable pair is acquainted (i.e. belongs to G).

Other parameters that can be varied (but were kept fixed) include:

1. The minimum and maximum lengths of men and women's preference lists (both fixed at 10) and
2. The popularity of the residents and hospitals.

We also ran the extended version of the Gale-Shapley algorithm (`Alg-gs`) on the generated instances. We measured the runtime taken by the algorithms and IP encoding as well as the size of the matchings obtained. Experiments were carried out on a Windows machine with 4 Intel(R) Core(R) i5-2400 CPUs at 3.1GHz and 8GB RAM.

4.7.2 Varying instance size

When the preference list lengths are kept constant, the number of men and women in the underlying SMI instance is an accurate indication of the size of the SMISS instance generated. We investigated the effect of varying SMISS instances sizes on the runtime and sizes of the matchings obtained. This was achieved by generating 10,000 random HRSS instances for each value of n_1 for the range $100, 200, ..., 1000$ with each instance having $n_1 = n_2$ and $d_G = 0.5$.

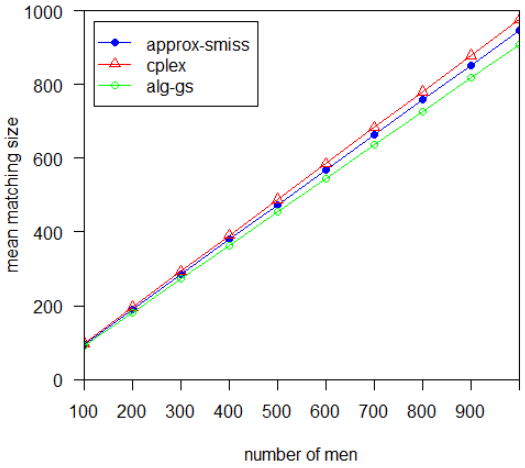

Figure 4.4: Mean matching size vs n_1

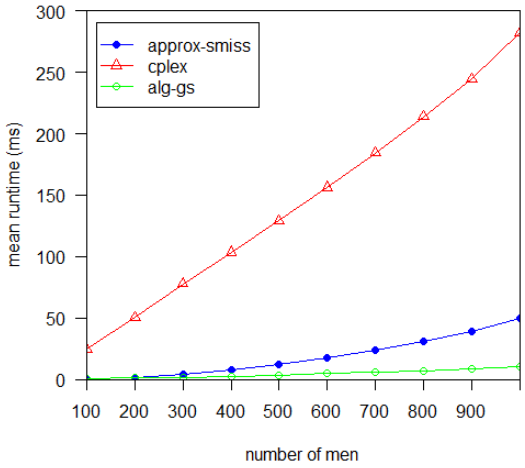

Figure 4.5: Mean runtime vs n_1

Figures 4.5 and 4.4 and Tables 4.1 and 4.2 summarise the results obtained. Figure 4.4 shows how the mean size of the matchings produced varies with n_1. As would be expected, `Alg-gs` produces the smallest matchings (which are stable in the classical sense) with `Approx-smiss` producing bigger (socially) stable matchings which are considerably larger than two-thirds of optimal as guaranteed by Theorem 4.5.6. Figure 4.5 shows the mean time it takes to solve an SMISS instance as we vary n_1. As shown in Table 4.1, both algorithms do considerably better than CPLEX with the `Alg-gs` performing the best. This is likely to be due to the exponential nature of the underlying branch-and-bound search algorithm employed by CPLEX in solving IP encodings and the fact that CPLEX is solving these problems to optimality.

4.7.3 Varying the density of the social network

In this subsection we investigate the effect of varying the density of the social network on the runtime and sizes of the matchings obtained. We varied d_G from 0 to 1 by steps of size 0.1. As mentioned earlier, MAX HRSS (and MAX SMISS) becomes polynomially solvable when $d_G = 0$ (this reduces to finding a maximum matching in a bipartite graph) or when $d_G = 1$ (this reduces to finding a stable matching in the underlying

n_1	Alg-gs	Approx-smiss	CPLEX
100	0.54	0.49	24.37
200	1.23	1.82	50.46
300	1.83	4.35	77.79
400	2.63	7.70	103.46
500	3.54	12.01	129.07
600	4.57	17.25	156.27
700	5.71	23.48	184.29
800	6.96	30.92	213.81
900	8.30	39.33	244.43
1000	9.91	49.91	281.88

Table 4.1: Mean runtime (ms) produced by both algorithms and CPLEX

n_1	Alg-gs	Approx-smiss	CPLEX
100	91.21 (93.35%)	95.05 (97.28%)	97.71 (100%)
200	182.06 (93.22%)	189.76 (97.17%)	195.29 (100%)
300	272.89 (93.17%)	284.54 (97.15%)	292.90 (100%)
400	363.76 (93.16%)	379.28 (97.13%)	390.48 (100%)
500	454.54 (93.15%)	473.97 (97.13%)	487.98 (100%)
600	545.38 (93.12%)	568.79 (97.12%)	585.65 (100%)
700	636.25 (93.14%)	663.43 (97.12%)	683.11 (100%)
800	727.04 (93.13%)	758.16 (97.11%)	780.71 (100%)
900	808.06 (92.00%)	853.00 (97.11%)	878.36 (100%)
1000	908.83 (93.13%)	947.68 (96.81%)	975.85 (100%)

Table 4.2: Mean matching sizes produced by both algorithms and CPLEX

HR/SMISS instance). We tested the effect of varying d_G by generating $10,000$ random SMI instances for each value of d_G with each instance having $n_1 = n_2 = 500$.

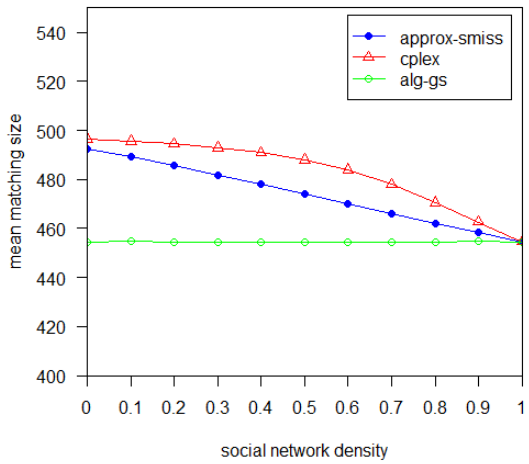

Figure 4.6: Mean matching size vs d_G

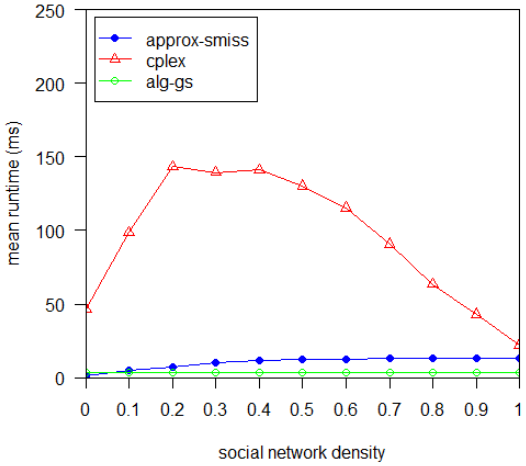

Figure 4.7: Mean runtime vs d_G

Figures 4.6 and 4.7 show the results obtained. As expected `Alg-gs` is constant in both figures as it does not depend on G. In terms of the runtime shown in Figure 4.7, `Approx-smiss` performs well with the runtime rising only slightly across the entire range. The CPLEX solver however takes a considerably longer time to find an optimal solution. The runtime curve is somewhat bell-shaped with its highest region between $d_G = 0.2$ and $d_G = 0.5$ for the instances tested. It is at this range the solver encounters the most challenging instances. For values of $d_G > 0.5$ the increasing density of G means an increase in the number of stability-enforcing constraints thus making it easier for the solver to rule out potential solutions. This trend continues until every acceptable pair is an acquainted one. At this point ($d_G = 1$) results show that CPLEX finds it easier to reach an optimal solution (a stable matching) than when $d_G = 0$ (a maximum matching). For values of $d_G < 0.2$ the relatively small amount of potential socially blocking pairs would mean optimal solutions are likely to be very close to complete matchings (as observed in Figure 4.7). This makes it a lot easier for the CPLEX solver to terminate the search process once it obtains a feasible solution that has all n_1 men matched.

Figure 4.7 shows how the sizes of the matchings obtained varies as we increase d_G. Once again `Alg-gs` produces matchings whose sizes are independent of d_G. The mean size of the matchings obtained by both the `Approx-smiss` algorithm and CPLEX falls gradually as we increase d_G, with both curves converging to meet the the `Alg-gs` curve at $d_G = 1$. We observe that the `Approx-smiss` algorithm performs best at the extremes of the range with the widest gap between the mean sizes of its matchings and the optimal solutions occurring at the middle point ($d_G = 0.4 - 0.6$). The large area between the `Alg-gs` curve and the other two is an indication of the benefit that stands to be gained by adopting this relaxed notion of stability thus justifying our investigation of social stability.

4.8 Conclusion

In this chapter we presented a formal definition of HRSS as well as algorithmic results relating to MAX HRSS. An obvious open problem in this area involves extending the algorithm for $(2, \infty)$-MAX SMISS to the HRSS case. Further, In Theorem 4.3.1 we showed a reduction from HRSS to HR+SN. However the converse statement of the theorem places a severe restriction on the HR+SN instances (they must admit resident-complete locally stable matchings). It remains to be shown that a reduction exists from HRSS to HR+SN that does not require such a restriction.

Although this chapter focused on optimising the size of socially stable matchings, other

optimality criteria may also be considered (subject to the size criteria). We can also extend the notion of social stability to other matching problems and their variants. Some of these directions are considered next in Chapter 5.

Chapter 5

Further Algorithmic Results on Socially Stable Matchings

5.1 Introduction

The notion of social stability in matching problems was discussed in Section 2.5. In Chapter 4 we presented the first set of algorithmic results relating to optimising the size of socially stable matchings in HRSS instances (MAX HRSS). In this chapter we extend our investigation of social stability to other matching problems and considering other optimality criteria.

We begin by considering some fundamental structural properties of stable matchings in SMI that do not carry over to the case of socially stable matchings. An instance of SMISS does not in general admit a lattice structure for the set of socially stable matchings. Indeed, there can be no guarantee of the existence of a man-optimal (or woman-optimal) socially stable matching. Figure 5.1 shows an SMISS instance where there is no analogue of the man-oriented dominance relation between stable matchings in SM [38] for the two socially stable matchings $M_1 = \{(m_1, w_1), (m_2, w_2)\}$ and $M_2 = \{(m_1, w_2), (m_2, w_1)\}$. Also Figure 5.2 shows an SMISS instance, with socially stable matchings $M_1 = \{(m_1, w_1)\}$ and $M_2 = \{(m_1, w_2), (m_2, w_1)\}$, in which the Rural Hospital's Theorem fails on two counts: (i) m_2 is unmatched in M_1 but is matched in M_2, and (ii) w_2 is undersubscribed in M_1 but is full in M_2.

The rest of the chapter is structured as follows. In Section 5.2, we present hardness and approximability results for the problems of finding "fair" socially stable matchings. The optimality problems considered include finding egalitarian, minimum regret and sex-equal socially stable matchings. In Section 5.3 we consider other variants of COM SMISS that are of practical relevance or theoretical significance. We consider cases

men's preferences	women's preferences
$m_1 : w_1 \quad w_2$	$w_1 : m_1 \quad m_2$
$m_2 : w_1 \quad w_2$	$w_2 : m_1 \quad m_2$

empty social network graph (i.e. $G = \emptyset$)

Figure 5.1: No man-optimal socially stable matching in SMISS instance (I, G)

men's preferences	women's preferences
$m_1 : w_1 \quad w_2$	$w_1 : m_1 \quad m_2$
$m_2 : w_1$	$w_2 : m_1$

empty social network graph (i.e. $G = \emptyset$)

Figure 5.2: Rural Hospitals Theorem fails in SMISS instance (I, G)

where the degree and structure of the social network is restricted. We also consider cases where the there are restrictions on the ordering of the agents' preference lists. We show NP-completeness for COM SMISS in most of the cases considered. In Section 5.4 we show that MIN SMISS-D, the problem of deciding whether an SMISS instance admits a socially stable matching of size $\leq k$ where k is some integer, is NP-complete. Although we do not identify potential practical applications for this problem, it is still of considerable theoretical interest. Section 5.5 presents further hardness results related to determining whether a given agent, or a given pair is matched in a socially stable matching in the SM or HR domains. We move from two-sided to one-sided matching problems in Section 5.6 and consider social stability in the roommates context. We present hardness and inapproximability results that are analogous to those presented for the SM and HR cases. The chapter ends with a brief conclusion in Section 5.7.

5.2 Fair socially stable matchings

5.2.1 Introduction

It is natural in some real-world applications to seek socially stable matchings that are *fair* to both sets of agents. Various criteria for measuring this fairness of stable matchings have been investigated in the case of stable marriage problems involving ties [39]. In this section we extend these ideas to the HRSS context.

We define the cost of a matching M for an agent a_i as the position of $M(a_i)$ on a_i's list, assuming a_i is matched in M. The cost $c(M)$ of a matching M, in the SMSS context, is derived below:

$c_{a_i}(M) = rank(a_i, M(a_i))$ where a_i is matched in M.

$c_{\mathcal{U}}(M) = \sum_{m \in \mathcal{U}} c_m(M)$ where $\mathcal{U}$ is the set men in the instance.

$c_{\mathcal{W}}(M) = \sum_{w \in \mathcal{W}} c_w(M)$ where $\mathcal{W}$ is the set women in the instance.

$c(M) = c_{\mathcal{U}}(M) + c_{\mathcal{W}}(M)$

We define the *regret* of a matching M as the highest cost of M for an agent taken over all agents matched in M. Thus

$$r(M) = \max\{c_{a_i}(M), \text{ where } a_i \text{ is matched in } M\}$$

We define the *Stable Marriage problem under Social Stability (*SMSS*)* as a special case of SMISS in which $n_1 = n_2$ and all preference lists are complete. An instance of SMSS can admit socially stable matchings of different sizes but a maximum socially stable matching (of size n_1) can be found in polynomial time by simply finding a stable matching in the underlying SM instance using the Gale-Shapley algorithm. This is because all preference lists are complete in an SMSS instance and stable matchings in this context are complete. The objective is to find a complete socially stable matching satisfying some additional optimality criteria relating to the cost to the agents involved. In this section we consider three such criteria, egalitarian, minimum regret and sex equal socially stable matchings. We restrict our investigation to the SMSS case where matchings are required to be complete.

5.2.2 Egalitarian socially stable matchings

Given an instance (I, G) of SMSS, an *egalitarian socially stable matching* is a matching M whose cost $c(M)$ is minimum over all socially stable matchings in (I, G). We define EGALITARIAN SMSS to be the problem of finding an egalitarian socially stable matching given an SMSS instance. In this section, we show that it is NP-hard to approximate EGALITARIAN SMSS, to within a factor of $N^{1-\epsilon}$ where N is the number of men and $\epsilon > 1$. Let I be an SMTI instance where ties occur only in the men's preference lists and where each tie appears only at the end of a preference list. Then I can be reduced to an SMSS instance (I', G) in polynomial time such that an $N^{1-\epsilon}$-approximation algorithm for EGALITARIAN SMSS on (I', G) decides COM SMTI on I.

Theorem 5.2.1. *It is NP-hard to approximate* EGALITARIAN SMSS *to within a factor of $N^{1-\varepsilon}$ where N is the number of men involved and $\varepsilon > 0$.*

Proof. Let $\varepsilon > 0$ be given. We employ a reduction similar to the one used in the proof of Theorem 2 in [64] where it is shown that approximating MIN SMTI to within $N^{1-\varepsilon}$

is NP hard for any $\varepsilon > 0$. Here we reduce an instance I of $(3,3)$-COM SMTI (where the ties occur in the men's lists only, each tie occurs at the tail of some list and preference lists lengths are at most 3) to an instance (I', G) of EGALITARIAN SMSS.

Let $\mathcal{U} = \{m_1, m_2, ..., m_n\}$ and $\mathcal{W} = \{w_1, w_2, ..., w_n\}$ be the sets of men and women in I respectively. We construct (I', G) as follows: let $\mathcal{U}' = \{m'_1, m'_2, ..., m'_{n^c}\} \cup \bigcup_{1 \le j \le n^{c-1}} \mathcal{U}_j$ be the set of men where $\mathcal{U}_j = \{m_{j,1}, m_{j,2}, ..., m_{j,n}\}$ is a copy of $\mathcal{U}$ and $c = \lceil 2/\varepsilon \rceil$. Similarly let $\mathcal{W}' = \{w'_1, w'_2, ..., w'_{n^c}\} \cup \bigcup_{1 \le j \le n^{c-1}} \mathcal{W}_j$ be the set of women where $\mathcal{W}_j = \{w_{j,1}, w_{j,2}, ..., w_{j,n}\}$ is a copy of $\mathcal{W}$. There are $2n^c$ men and $2n^c$ women in I'. The preference lists of the men and women in (I', G) are constructed as follows.

$$
\begin{array}{ll}
m'_i : w'_i \ ... & (1 \le i \le n^c) \\
m_{j,i} : P_{j,i} \ \langle \text{all } w'_i \text{ in any order} \rangle \ ... & (1 \le i \le n, 1 \le j \le n^{c-1}) \\
w'_i : m'_i \ ... & (1 \le i \le n^c) \\
w_{j,i} : Q_{j,i} \ \langle \text{all } m'_i \text{ in any order} \rangle \ ... & (1 \le i \le n, 1 \le j \le n^{c-1})
\end{array}
$$

We use the symbol "..." at the end of each preference list to denote a list of the remaining people of opposite sex in arbitrary order. We denote $P_{j,i}$ and $Q_{j,i}$ as preference lists consisting of women and men in the j'th clone of $\mathcal{U}$ and $\mathcal{W}$ respectively. The preference ordering is derived from the preference lists of m_i and w_i in I respectively. For each j $(1 \le j \le n^{c-1})$ we add the pair $(m_{j,p}, w_{j,q})$ to A if w_q does not appear in a tie on m_p's preference list in I. This completes the construction of (I', G). It is easy to see that for each i $(1 \le i \le n^c)$, (m'_i, w'_i) will be matched in any socially stable matching in (I', G).

If I admits a complete stable matching M then construct a socially stable matching M' in (I', G) as follows:

$$M' \ = \ \{(m'_i, w'_i) : 1 \le i \le n^c\} \ \cup \ \{(m_{j,p}, w_{j,q}) : (m_p, w_q) \in M \ \wedge \ 1 \le j \le n^{c-1}\}$$

It is easy to see that M' is socially stable in (I', G) and since every man and woman in M is matched to their third choice partner or better, we can estimate an upper bound for $c(M')$ as follows:

$$
\begin{array}{rcl}
c(M') & \le & 2(n^c + n^{c-1} \times n \times 3) \\
& \le & 2(n^c + 3n^c) \\
& \le & n^{c+1}
\end{array}
$$

if we assume without loss of generality that $n \ge 8$.

On the other hand, if I does not admit a complete stable matching then for any

complete socially stable matching M' in (I', G) at least one man and one woman in M' is matched to their $(1 + n^c + 1)$th partner or worse. We can estimate a lower bound for $c(M')$ as follows:

$$\begin{aligned} c(M') &\geq 2 \times (n^c + n^{c-1} \times (n-1) + n^{c-1} \times (1 + n^c + 1)) \\ &\geq 4n^c + 2n^{2c-1} + 2n^{c-1} \\ &> 2n^{2c-1} \end{aligned}$$

Let $N(= 2n^c)$ denote the number of men in (I', G). Now assume that EGALITARIAN SMSS has an $N^{1-\varepsilon}$-approximation algorithm $\mathtt{A}$. If I admits a complete stable matching, $\mathtt{A}$ returns a socially stable matching M' such that $c(M') \leq n^{c+1}N^{1-\varepsilon}$ where $n^{c+1}N^{1-\varepsilon} \leq n^{c+1}(2n^c)^{1-2/c} = 2^{1-c/2}n^{2c-1}$. Thus $c(M') \leq 2n^{2c-1}$.

Also if I admits no complete stable matching then J returns a socially stable matching M' such that $c(M') > 2n^{2c-1}$. Hence the existence of J would allow $(3,3)$-COM SMTI to be solved in polynomial time, a contradiction unless P=NP. □

5.2.3 Minimum regret socially stable matchings

Given an instance (I, G) of SMSS, a *minimum regret socially stable matching* is a matching with the lowest regret taken over all complete socially stable matchings in (I, G). We define MIN REGRET SMSS to be the problem of finding a minimum regret socially stable matching given an SMSS instance. In this subsection, we show that this problem is NP-hard to approximate to within a factor of δN where N is the number of men and $\delta < 1/3$.

It is trivial to observe that, given an SMSS instance, for any complete socially stable matching M, $1 \leq r(M) \leq n$. Thus an n-approximation for MIN REGRET SMSS can be found in polynomial time. We now show the inapproximability result by reducing from an instance of $(3,3)$-COM SMTI.

Theorem 5.2.2. *It is NP-hard to approximate* MIN REGRET SMSS *to within a factor of* $\frac{1}{3}N$ *where* N *is the number of men involved.*

Proof. We consider an instance I of the NP-complete problem $(3,3)$-COM SMTI [61] where the ties occur in the men's lists only, each tie occurs at the tail of some list and preference lists are of length at most 3. Let $\mathcal{U} = \{m_1, m_2, ..., m_n\}$ and $\mathcal{W} = \{w_1, w_2, ..., w_n\}$ be the sets of men and women in I respectively. We construct an SMSS instance (I', G) as follows: let the men and women in I' be $\mathcal{U} \cup \mathcal{U}'$ and $\mathcal{W} \cup \mathcal{W}'$ respectively, where $\mathcal{U}' = \{m'_1, m'_2, ..., m'_R\}$, $\mathcal{W}' = \{w'_1, w'_2, ..., w'_R\}$ and $R = 3\delta n/(1-3\delta)$ for some constant $0 < \delta < 1/3$. The preference lists of the men and women in I' are constructed below.

m_i : ⟨preference list of m_i in I⟩ ⟨all w'_j in any order⟩ ... $(1 \leq i \leq n)$

m'_i : w'_i ... $(1 \leq i \leq R)$

w_j : ⟨preference list of w_j in I⟩ ⟨all m'_i in any order⟩ ... $(1 \leq j \leq n)$

w'_j : m'_j ... $(1 \leq j \leq R)$

We use the symbol "..." at the end of each preference list to denote a list of the remaining people of opposite sex in arbitrary order. We add the pair (m_i, w_j) to A if w_j does not appear in a tie on m_i's preference list in I.

Suppose that M is a complete stable matching in I. Then a socially stable matching M' in (I', G) can be constructed as follows: $M' = M \cup \{(m'_i, w'_i) : 1 \leq i \leq R\}$. Thus $r(M') \leq 3$. Conversely suppose I does not admit a complete stable matching in I. Then for any complete socially stable matching M' in (I', G), $r(M') \geq 1 + R + 1 > R$.

Let $N(= R + n)$ denote the number of men in (I', G). Then $N = n/(1 - 3\delta)$ so $N - n = 3\delta N$, i.e. $R = 3\delta N$. Now assume that MIN REGRET SMSS has a δN-approximation algorithm J. If I admits a complete stable matching, A returns a socially stable matching M' such that $r(M') \leq 3\delta N$. Also if I admits no complete stable matching then J returns a socially stable matching M' such that, given our choice of R, $r(M') > R = 3\delta N$. Hence the existence of J would allow $(3, 3)$-COM SMTI to be solved in polynomial time, a contradiction unless P=NP. □

5.2.4 Sex-equal socially stable matchings

Given an instance (I, G) of SMSS, a *sex-equal socially stable matching* is a socially stable matching M in which the difference $d(M) = |c_{\mathcal{U}}(M) - c_{\mathcal{W}}(M)|$ is a minimum. We define SEX EQUAL SMSS to be the problem of finding a sex-equal socially stable matching given an SMSS instance. Once again we establish an inapproximability result for the problem by reducing from an instance of $(3, 3)$-COM SMTI.

Theorem 5.2.3. *It is NP-hard to approximate* SEX EQUAL SMSS *to within a constant factor c.*

Proof. We consider an instance I of the NP-complete problem $(3, 3)$-COM SMTI [61] where the ties occur in the men's lists only, each tie occurs at the tail of some list and preference lists are of length at most 3. Let $\mathcal{U} = \{m_1, m_2, ..., m_n\}$ and $\mathcal{W} = \{w_1, w_2, ..., w_n\}$ be the sets of men and women in I respectively. We construct an SMSS instance (I', G) as follows: let the men and women in I' be $\mathcal{U} \cup \mathcal{U}' \cup \mathcal{U}''$ and $\mathcal{W} \cup \mathcal{W}' \cup \mathcal{W}''$ respectively, where $\mathcal{U}' = \{m'_1, m'_2, ..., m'_{R'}\}$, $\mathcal{U}'' = \{m''_1, m''_2, ..., m''_{R''}\}$, $\mathcal{W}' =$

$\{w'_1, w'_2, ..., w'_{R'}\}$, $\mathcal{W}'' = \{w''_1, w''_2, ..., w''_{R''}\}$, $R' = n$ and $R'' = \lceil c \rceil n^2$. The preference lists of the men and women in I' are constructed below.

$$
\begin{array}{ll}
m_i : P_i \;\; \mathcal{W}\backslash P_i \;\; \mathcal{W}' \;\; \mathcal{W}'' & (1 \leq i \leq n) \\
m'_i : w'_i \; ... & (1 \leq i \leq R') \\
m''_i : w''_i \; ... & (1 \leq i \leq R'') \\
w_j : \mathcal{U}' \;\; Q_j \;\; \mathcal{U}'' \;\; \mathcal{U}\backslash Q_j & (1 \leq j \leq n) \\
w'_j : m'_j \; ... & (1 \leq j \leq R') \\
w''_j : m''_j \; ... & (1 \leq j \leq R'')
\end{array}
$$

We use the symbol "..." to denote a list of the remaining people of opposite sex in arbitrary order. We denote P_i and Q_j to be the preference lists of m_i and w_j in I respectively. Also where the symbols $\mathcal{W}', \mathcal{W}'', \mathcal{U}'$ and $\mathcal{U}''$ appear, the elements of these sets are listed in arbitrary order in the preference lists. We add the pair (m_i, w_j) to A where $A = E(G)$ if w_j does not appear in a tie on m_i's preference list in I.

Suppose that M is a complete stable matching in I. Then a socially stable matching M' in (I', G) can be constructed as follows:

$$M' = M \cup \{(m'_i, w'_i) : 1 \leq i \leq R'\} \cup \{(m''_i, w''_i) : 1 \leq i \leq R''\}$$

Since every man and woman in M is matched to their third choice partner or better, we can estimate an upper bound for $d(M')$ as follows:

$$
\begin{array}{rcl}
c_{\mathcal{U}}(M') & \leq & 3n + R' + R'' \\
c_{\mathcal{W}}(M') & \leq & n(R' + 3) + R' + R'' \\
d(M') & = & |c_{\mathcal{W}}(M) - c_{\mathcal{U}}(M)| \\
 & \leq & n_1 R' \\
 & = & n^2
\end{array}
$$

Conversely suppose I does not admit a complete stable matching in I. Then for any complete socially stable matching M' in (I', G) at least one woman in M' is matched to her $(R' + 3 + R'' + 1)$th partner or worse. Thus:

$$
\begin{array}{rcl}
c_{\mathcal{U}}(M') & \leq & n^2 + R' + R'' \\
c_{\mathcal{W}}(M') & \geq & (R' + 3 + R'' + 1) + (n-1)(R' + 1) + R' + R'' \\
d(M') & = & |c_{\mathcal{W}}(M) - c_{\mathcal{U}}(M)| \\
 & \geq & (R' + R'' + 4) + (n-1)(R' + 1) - n^2 \\
 & \geq & 3 + R'' + nR' + n - n^2 \\
 & > & cn^2
\end{array}
$$

Now assume that SEX EQUAL SMSS has a c-approximation algorithm A. If I admits a complete stable matching, A returns a socially stable matching M' such that $d(M') \leq cn^2$. Also if I admits no complete stable matching then A returns a socially stable matching M' such that $d(M') > cn^2$. Hence the existence of A would allow $(3,3)$-COM SMTI to be solved in polynomial time, a contradiction unless P=NP. □

5.3 Further restrictions of COM SMISS

In this section we investigate other variants of COM SMISS that are of practical relevance or theoretical interest. We present results relating to the complexity of these problems. In Chapter 4 we investigated two restrictions of the problem based on the number of acquainted and unacquainted pairs and provided FPT algorithms for both cases. We consider variants with other restrictions on the social network graph as well as the underlying SM instance. In Section 5.3.1 we show COM SMISS to be NP-complete even under severe restrictions on the degree of the social network graph and the lengths of the preference lists. In Section 5.3.2 we consider restrictions to the structure of the social network graph. We define the notions of *fully acquainted* and *fully unacquainted* agents and we show COM SMISS to be NP-complete under these restrictions. Finally in Section 5.3.3 we consider the cases where preference lists are derived from so-called *master lists*. We we show COM SMISS to be NP-complete even when one or both sets of preference lists are derived from master lists.

5.3.1 Restrictions on the degree of the social network graph

We start by showing that COM SMISS is NP-complete even if each preference list is of length at most 3 and the social network graph has maximum degree 1. The reduction and subsequent arguments presented here will serve as a basis of other NP-hardness proofs in the following subsections. Our proof of this result uses a reduction from a restricted version of SAT. More specifically, let (2,2)-E3-SAT denote the problem of deciding, given a Boolean formula B in CNF in which each clause contains exactly 3 literals and, for each $v_i \in V$, each of literals v_i and $\bar{v}_i$ appears exactly twice in B, whether B is satisfiable. Berman et al. [14] showed that (2,2)-E3-SAT is NP-complete.

Theorem 5.3.1. COM SMISS *is NP-complete, even if each preference list is of length at most 3 and the social network graph has maximum degree 1.*

Proof. Let B be an instance of (2,2)-E3-SAT. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the sets of variables and clauses respectively in B. Then for each

$$
\begin{array}{rll}
x_{4i}: & y_{4i} \;\; c(x_{4i}) \;\; y_{4i+1} & (0 \le i \le n-1) \\
x_{4i+1}: & y_{4i+1} \;\; c(x_{4i+1}) \;\; y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+2}: & y_{4i+3} \;\; c(x_{4i+2}) \;\; y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+3}: & y_{4i} \;\; c(x_{4i+3}) \;\; y_{4i+3} & (0 \le i \le n-1) \\
p_j^s: & z_j \;\; c_j^s & (1 \le j \le m \wedge 1 \le s \le 3) \\
q_j: & c_j^1 \;\; c_j^2 \;\; c_j^3 & (1 \le j \le m)
\end{array}
$$

$$
\begin{array}{rll}
y_{4i}: & x_{4i} \;\; x_{4i+3} & (0 \le i \le n-1) \\
y_{4i+1}: & x_{4i} \;\; x_{4i+1} & (0 \le i \le n-1) \\
y_{4i+2}: & x_{4i+1} \;\; x_{4i+2} & (0 \le i \le n-1) \\
y_{4i+3}: & x_{4i+2} \;\; x_{4i+3} & (0 \le i \le n-1) \\
c_j^s: & p_j^s \;\; x(c_j^s) \;\; q_j & (1 \le j \le m \wedge 1 \le s \le 3) \\
z_j: & p_j^1 \;\; p_j^2 \;\; p_j^3 & (1 \le j \le m)
\end{array}
$$

Figure 5.3: Preference lists in the constructed instance of COM SMISS.

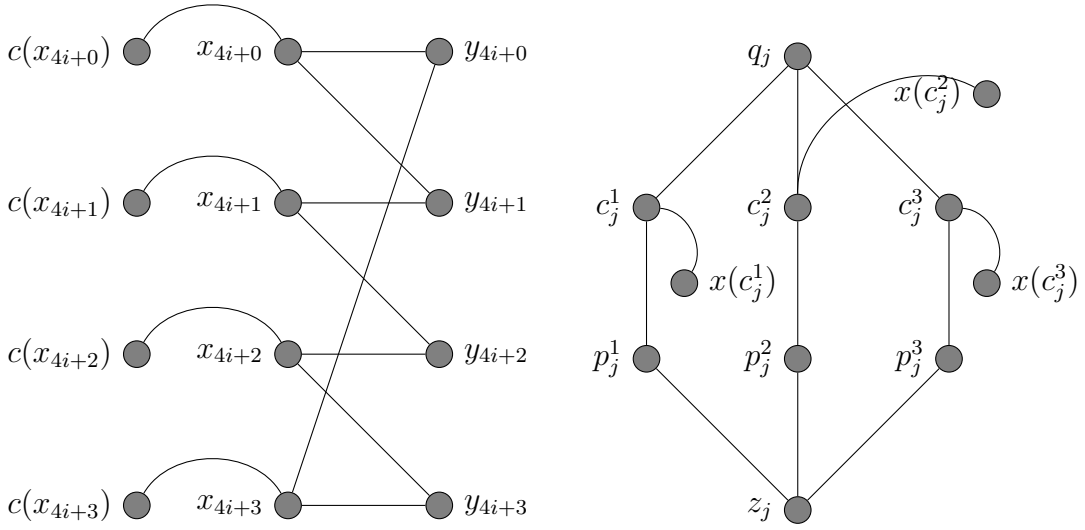

Figure 5.4: Pictorial representation of the preference lists.

$v_i \in V$, each of literals v_i and $\bar{v_i}$ appears exactly twice in B. (Hence $m = \frac{4n}{3}$.) Also $|c_j| = 3$ for each $c_j \in C$. The idea is to construct an instance (I, G) of COM SMISS from B such that each literal in B, each position in B and each cluster in C corresponds to a man and a woman in I. The preference lists and social network are constructed to enable us show that B admits a satisfying truth assignment if and only if a complete socially stable matching in (I, G) can be formed. We generate (I, G) as follows. The set of men in I is $X \cup P \cup Q$, where $X = \cup_{i=0}^{n-1} X_i$, $X_i = \{x_{4i+r} : 0 \le r \le 3\}$ $(0 \le i \le n-1)$, $P = \cup_{j=1}^{m} P_j$, $P_j = \{p_j^1, p_j^2, p_j^3\}$ $(1 \le j \le m)$ and $Q = \{q_j : c_j \in C\}$. The set of women in I is $Y \cup C' \cup Z$, where $Y = \cup_{i=0}^{n-1} Y_i$, $Y_i = \{y_{4i+r} : 0 \le r \le 3\}$ $(0 \le i \le n-1)$, $C' = \{c_j^s : c_j \in C \wedge 1 \le s \le 3\}$ and $Z = \{z_j : c_j \in C\}$.

The preference lists of the men and women in I are shown in Figures 5.3 and 5.4. In the preference list of an agent $x_{4i+r} \in X$ ($0 \le i \le n-1$ and $r \in \{0, 1\}$), the symbol $c(x_{4i+r})$

denotes the woman $c_j^s \in C'$ such that the $(r+1)$th occurrence of literal v_i appears at position s of c_j. Similarly if $r \in \{2,3\}$ then the symbol $c(x_{4i+r})$ denotes the woman $c_j^s \in C'$ such that the $(r-1)$th occurrence of literal $\bar{v}_i$ appears at position s of c_j. Also in the preference list of an agent $c_j^s \in C'$, if literal v_i appears at position s of clause $c_j \in C$, the symbol $x(c_j^s)$ denotes the man x_{4i+r-1} where $r = 1, 2$ according as this is the first or second occurrence of literal v_i in B. Otherwise if literal $\bar{v}_i$ appears at position s of clause $c_j \in C$, the symbol $x(c_j^s)$ denotes the man x_{4i+r+1} where $r = 1, 2$ according as this is the first or second occurrence of literal $\bar{v}_i$ in B. Clearly each preference list is of length at most 3.

The social network graph G is constructed as follows: the vertices are $X \cup C'$, and the edges are $\{x_k, c(x_k)\}$ $(0 \leq k \leq 4n-1)$. Clearly no two edges in G are adjacent, and thus G has maximum degree 1. For each i $(0 \leq i \leq n-1)$, let $T_i = \{(x_{4i+r}, y_{4i+r}) : 0 \leq r \leq 3\}$ and $F_i = \{(x_{4i+r}, y_{4i+r+1})\} : 0 \leq r \leq 3\}$, where addition is taken modulo 4. A maximum socially stable matching M in I would contain either T_i $(0 \leq i \leq n-1)$ or F_i $(0 \leq i \leq n-1)$. Our strategy is to show that, in both cases, the existence of M would mean that B is satisfiable and vice versa.

We claim that B is satisfiable if and only if I admits a complete socially stable matching.

For, let f be a satisfying truth assignment of B. Define a complete matching M in I as follows. For each variable $v_i \in V$, if v_i is true under f, add the pairs in T_i to M, otherwise add the pairs in F_i to M. Now let $c_j \in C$. As c_j contains a literal that is true under f, let $s \in \{1,2,3\}$ denote the position of c_j in which this literal occurs. Add the pairs (p_j^t, c_j^t) $(1 \leq t \neq s \leq 3)$, (p_j^s, z_j) and (q_j, c_j^s) to M.

Now suppose that $(x_{4i+r}, c(x_{4i+r}))$ socially blocks M, where $0 \leq i \leq n-1$ and $0 \leq r \leq 3$. Let $c_j^s = c(x_{4i+r})$, where $1 \leq j \leq m$ and $1 \leq s \leq 3$. Then $(q_j, c_j^s) \in M$. If $r \in \{0,1\}$ then $(x_{4i+r}, y_{4i+r+1}) \in M$, so that v_i is false under f. But literal v_i occurs in c_j, a contradiction, since literal v_i was supposed to be true under f by construction of M. Hence $r \in \{2,3\}$ and $(x_{4i+r}, y_{4i+r}) \in M$, so that v_i is true under f. But literal $\bar{v}_i$ occurs in c_j, a contradiction, since literal $\bar{v}_i$ was supposed to be true under f by construction of M. Hence M is socially stable in I.

Conversely suppose that M is a complete socially stable matching in I. We form a truth assignment f in B as follows. For each i $(0 \leq i \leq n-1)$, $M \cap (X_i \times Y_i)$ is a perfect matching of $X_i \cup Y_i$. If $M \cap (X_i \times Y_i) = T_i$, set v_i to be true under f. Otherwise $M \cap (X_i \times Y_i) = F_i$, in which case we set v_i to be false under f.

Now let c_j be a clause in C $(1 \leq j \leq m)$. There exists some s $(1 \leq s \leq 3)$ such that $(q_j, c_j^s) \in M$. Let $x_{4i+r} = x(c_j^s)$ for some i $(0 \leq i \leq n-1)$ and r $(0 \leq r \leq 3)$. If $r \in \{0,1\}$ then $(x_{4i+r}, y_{4i+r}) \in M$ by the social stability of M. Thus variable v_i is true under f, and hence clause c_j is true under f, since literal v_i occurs in c_j. If $r \in \{2,3\}$

then $(x_{4i+r}, y_{4i+r+1}) \in M$ (where addition is taken modulo 4) by the social stability of M. Thus variable v_i is false under f, and hence clause c_j is true under f, since literal $\bar{v}_i$ occurs in c_j. Hence f is a satisfying truth assignment of B. □

5.3.2 Restrictions on the structure of the social network graph

Another interesting restriction on the social network graph is the case where an agent is either acquainted with all the agents on his/her preference list or with none of them. Such a scenario can arise in the context of assigning children to school places. Some schools involved in the matching schemes may be less concerned about stability and accept their assigned students while others may be more likely to abandon the matching for preferred students if offered. In the context of social stability, we can consider all the schools in the former case as *fully unacquainted* with the students on their list and the schools in the latter case as *fully acquainted* with the students on their list. We investigate two variants of this model in the SM context.

Define an agent a to be *fully acquainted* (respectively *fully unacquainted*) if a forms an acquainted (respectively unacquainted) pair with every agent on his/her/its list. In our first variant, each agent is fully acquainted or fully unacquainted. We remark that if an agent a is fully acquainted (respectively unacquainted) then so is every agent on a's preference list. Considering an SMISS instance (I, G) with a such restriction, the underlying SMI instance can then be partitioned into two sub-instances I' and I'' of SMI with A and U corresponding to the set of acceptable pairs in I' and I'' respectively.

In this variant, socially stable matchings can be of different sizes, as any stable matching in I' can be added to any element of the set of all matchings in I'' (including the empty set) to yield a socially stable matching in I. However finding a maximum socially stable matching can be done efficiently. Simply find a stable matching in I' and union that with a maximum matching in I''. It is straightforward to verify that the resulting matching will be a maximum socially stable matching in I.

In the second variant (a more general case), we assume that only the members of one set of agents (say the women) are required to be fully acquainted or fully unacquainted. In this case we are unable to partition the underlying SMI instance into two disjoint sets because if a woman is fully acquainted, a man m_i on her list still could be unacquainted with a woman on his list. Socially stable matchings can also be of varying sizes in this context. This can be demonstrated using a similar method to the one outlined in the first variant. We now show that COM SMISS is NP-complete in this case.

Theorem 5.3.2. COM SMISS *is NP-complete, even if preference lists are at most 3 and every woman is either fully acquainted or fully unacquainted.*

Proof. We start by considering the reduction shown in Subsection 5.3.1 from (2,2)-E3-SAT to COM SMISS. We describe a reduction from any instance B of (2,2)-E3-SAT to an instance (I, G) of COM SMISS with the same set of men and women and the same preference lists as those of the instance shown in Figure 5.3. We define the social network graph G by making all women c_j^s (for all $1 \leq j \leq m$ and $1 \leq s \leq 3$) fully acquainted with all other women fully unacquainted. Thus $A = \{(p_j^s, c_j^s), (q_j, c_j^s), (x(c_j^s), c_j^s) : 1 \leq j \leq m \wedge 1 \leq s \leq 3\}$.

We claim that B is satisfiable if and only if I admits a complete socially stable matching.

Considering the first direction, let f be a satisfying truth assignment of B. We define a complete matching in M in the same way it is defined in the proof of Theorem 5.3.1. Although all c_j^s are fully acquainted in G for $1 \leq j \leq m$ and $1 \leq s \leq 3$, only $(x_{4i+r}, c(x_{4i+r}))$ can socially block M where $0 \leq i \leq n-1$ and $0 \leq r \leq 3$. The pairs (q_j^s, c_j^s) cannot block M in the classical sense because q_j^s appears at the tail of c_j^s's preference list for all $1 \leq j \leq m$ and $1 \leq s \leq 3$. Also (p_j^s, c_j^s) cannot block M in the classical sense because c_j^s appears at the tail of p_j^s's preference list for all $1 \leq j \leq m$ and $1 \leq s \leq 3$. The rest of the arguments presented in Theorem 5.3.1 follow naturally.

Considering the converse case, the same argument presented in the converse direction of Theorem 5.3.1 holds. The set of potential socially blocking pairs remains $\{(x_{4i+r}, c(x_{4i+r})) : 0 \leq i \leq n-1 \wedge 0 \leq r \leq 3\}$ and the converse argument presented in Theorem 5.3.1 follows naturally. □

5.3.3 Restrictions on the ordering of preference lists

Restrictions of COM SMISS based on the lengths of the preference lists have been considered in Section 4.6.1. In this section we consider the likely scenario where one or both sets of preference lists are derived from a global ranking of the concerned agents. For example, hospitals may derive their preference lists from the academic performance (expressed as a numerical score) of the residents it finds acceptable. Thus the total set of residents can be ordered via a *master list* from which hospitals derive the order of their preferences over their acceptable residents. We adopt this notion in the social stability context and define two variants of the problem.

In the first variant we enforce the existence of a master list on one set of preference lists only (i.e., either men or women) and let the agents of the other set specify their preference lists arbitrarily. We denote the problem as COM SMISS-1ML. We present the following theorem concerning the computational complexity of this problem.

Theorem 5.3.3. COM SMISS-1ML *is NP-complete.*

$$
\begin{array}{rll}
x_{4i}: & y_{4i} \;\; c(x_{4i}) \;\; y_{4i+1} & (0 \le i \le n-1) \\
x_{4i+1}: & y_{4i+1} \;\; c(x_{4i+1}) \;\; y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+2}: & y_{4i+3} \;\; c(x_{4i+2}) \;\; y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+3}: & y_{4i} \;\; c(x_{4i+3}) \;\; y_{4i+3} & (0 \le i \le n-1) \\
p_j^s: & z_j \;\; c_j^s & (1 \le j \le m \wedge 1 \le s \le 3) \\
q_j^s: & c_j^s \;\; t_j^1 \;\; t_j^2 & (1 \le j \le m \wedge 1 \le s \le 3) \\
\\
y_{4i}: & x_{4i} \;\; x_{4i+3} & (0 \le i \le n-1) \\
y_{4i+1}: & x_{4i} \;\; x_{4i+1} & (0 \le i \le n-1) \\
y_{4i+2}: & x_{4i+1} \;\; x_{4i+2} & (0 \le i \le n-1) \\
y_{4i+3}: & x_{4i+2} \;\; x_{4i+3} & (0 \le i \le n-1) \\
c_j^s: & p_j^s \;\; x(c_j^s) \;\; q_j^s & (1 \le j \le m \wedge 1 \le s \le 3) \\
z_j: & p_j^1 \;\; p_j^2 \;\; p_j^3 & (1 \le j \le m) \\
t_j^s: & q_j^1 \;\; q_j^2 \;\; q_j^3 & (1 \le j \le m \wedge 1 \le s \le 2)
\end{array}
$$

Figure 5.5: Preference lists in the constructed instance of COM SMISS-2ML.

Proof. Once again we adopt the reduction used in the proof of Theorem 5.3.1 by modifying the generated COM SMISS instance I. We construct a master list consisting of all the men in I and can thus consider I as a COM SMISS-1ML instance. We define the master list $\mathcal{P} = \langle p_1^1 \;\; p_1^2 \;\; p_1^3 \;\; p_2^1 \;\; p_2^2 \;\; p_2^3 \;\; \ldots \; p_m^1 \;\; p_m^2 \;\; p_m^3 \;\; x_0 \;\; x_1 \;\; \ldots \;\; x_{n-1} \;\; q_1 \;\; q_2 \;\; \ldots \;\; q_m \rangle$. The rest of the proof for Theorem 5.3.1 will then follow naturally. □

In the second variant master lists are used to derive both sets of preference lists thus forming COM SMISS-2ML. It is again unlikely that this variant can be solved in polynomial time, as we now demonstrate.

Theorem 5.3.4. COM SMISS-2ML *is NP-complete.*

Proof. Similar arguments to those used in the proof of Theorem 5.3.1 hold for this proof. Firstly we modify the reduction from an instance B of (2,2)-E3-SAT used in the proof of Theorem 5.3.1. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the sets of variables and clauses respectively in B. Then for each $v_i \in V$, each of the literals v_i and $\bar{v_i}$ appears exactly twice in B. (Hence $m = \frac{4n}{3}$.) Also $|c_j| = 3$ for each $c_j \in C$. Next we form an instance I of COM SMISS-2ML as follows. The set of men in I is $X \cup P \cup Q$, where $X = \cup_{i=0}^{n-1} X_i$, $X_i = \{x_{4i+r} : 0 \le r \le 3\}$, $P = \cup_{j=1}^{m} P_j$, $P_j = \{p_j^1, p_j^2, p_j^3\}$ and $Q = \{q_j^1, q_j^2, q_j^3 \;:\; 1 \le j \le m\}$. The set of women in I is $Y \cup C' \cup Z \cup T$, where $Y = \cup_{i=0}^{n-1} Y_i$, $Y_i = \{y_{4i+r} : 0 \le r \le 3\}$, $C' = \{c_j^s : c_j \in C \wedge 1 \le s \le 3\}$, $Z = \{z_j : c_j \in C\}$ and $T = \{t_j^1, t_j^2 \;:\; 1 \le j \le m\}$.

The preference lists of the men and women in I are shown in Figure 5.5. We carry over the same definitions for $c(x_{4_i+r})$ and $x(c_j^s)$ from the proof of Theorem 5.3.1. Both sets of preference lists can be seen to be derived from master lists. We construct the

master lists of women and men denoted $\mathcal{P}_\mathcal{W}$ and $\mathcal{P}_\mathcal{M}$ as follows. Let

$$\mathcal{P}_\mathcal{W} = \langle z_1 \;\; z_2 \;\; ... \;\; z_m \;\; \mathcal{P}_0 \;\; \mathcal{P}_1 \;\; ... \;\; \mathcal{P}_{n-1} \; t_1^1 \;\; t_1^2 \;\; t_2^1 \;\; t_2^2 \;\; ... \;\; t_m^1 \;\; t_m^2 \rangle$$
$$\text{where } \mathcal{P}_i = \langle y_{4i} \;\; c(x_{4i}) \;\; y_{4i+1} \;\; c(x_{4i+1}) \;\; c(x_{4i+3}) \;\; y_{4i+3} \;\; c(x_{4i+2}) \;\; y_{4i+2} \rangle$$

It is easy to see how all the men's preference lists from Figure 5.5 can be derived from $\mathcal{P}_\mathcal{W}$.

$$\mathcal{P}_\mathcal{M} = \langle p_1^1 \;\; p_1^2 \;\; p_1^3 \;\; p_2^1 \;\; p_2^2 \;\; p_2^3 \;\; ... \;\; p_m^1 \;\; p_m^2 \;\; p_m^3 \;\; x_0 \;\; x_1 \;\; ... \;\; x_{n-1}$$
$$q_1^1 \;\; q_1^2 \;\; q_1^3 \;\; q_2^1 \;\; q_2^2 \;\; q_2^3 \;\; ..., \;\; q_m^1 \;\; q_m^2 \;\; q_m^3 \rangle$$

It is easy to see how all the women's preference lists from Figure 5.5 can be derived from $\mathcal{P}_\mathcal{M}$. The social network graph G is constructed as follows: the vertices are $X \cup C'$, and the edges are $\{x_k, c(x_k)\}$ $(0 \leq k \leq 4n-1)$.

We claim that B is satisfiable if and only if I admits a complete socially stable matching. Similar arguments to those used in the proof of Theorem 5.3.1 hold for the proof of this claim. The only difference is the construction of the complete socially stable matching from a satisfying truth assignment. In the proof of Theorem 5.3.3, we had $(q_j, c_j^s) \in M$ for each $c_j \in C$ where c_j contained a literal at position $s \in \{1,2,3\}$ that is true. Here we add (q_j^s, c_j^s) to M. For the other two men $q_j^{s'}$ and $q_j^{s''}$ (where $\{s, s', s''\} = \{1,2,3\}$), we assume without loss of generality that $s' < s''$. We add $(q_j^{s'}, t_j^1)$ and $(q_j^{s''}, t_j^2)$ to M, thus ensuring that M is a complete socially stable matching. The rest of the arguments presented in Theorem 5.3.1 follow naturally. □

5.4 Minimum socially stable matchings

In this section we investigate the problem MIN SMISS-D which is defined as follows. Given an SMISS instance (I, G) and an integer k, decide whether I admits a socially stable matching of size at most k. We reduce from the MINIMUM MAXIMAL MATCHING problem which is defined as follows. Given a graph G and a positive integer k, decide whether G admits a maximal matching M such that $|M| \leq k$. MINIMUM MAXIMAL MATCHING is known to be NP-complete even for bipartite graphs where every left hand vertex has degree 2 and every right hand vertex has degree of exactly 3 [41].

Theorem 5.4.1. MIN SMISS-D *is NP-complete.*

Proof. Let (G, k) be an instance of MINIMUM MAXIMAL MATCHING where every left hand vertex has degree 2 and every right hand vertex has degree at most 3. We define the sets of left hand and right hand side vertices in G as $U = \{u_1, u_2, ..., u_{n_1}\}$ and $W = \{w_1, w_2, ..., w_{n_2}\}$ respectively. For each vertex $u_i \in U$, let $w_{i,1}$ and $w_{i,2}$ be the

two neighbours of u_i in W. Without loss of generality if $w_{i,1} = w_r$ and $w_{i,2} = w_s$, suppose $r < s$. For each vertex $w_j \in W$, let $u_{j,1}$, $u_{j,2}$ and $u_{j,3}$ be the three neighbours of w_j in W. Without loss of generality if $u_{j,1} = u_r$, $u_{j,2} = u_s$ and $u_{j,3} = u_t$, suppose $r < s < t$. The idea is to construct an instance (I, G') of MIN SMISS from G such that the preference lists in I would be determined (in part) by the edges in G. We can then show that, for a specific value k', G has a maximal matching of size $\leq k$ if and only if (I, G') has a socially stable matching of size $\leq k'$.

We construct (I, G') from G as follows. The set of men in I is $U' \cup Z$ where $U' = \{u_i^1, u_i^2 : 1 \leq i \leq n_1\}$ and $Z = \{z_j^1, z_j^2, z_j^3 : 1 \leq j \leq n_2\}$. The set of women in I is $W' \cup X \cup Y$ where $W' = \{w_j^1, w_j^2, w_j^3 : 1 \leq j \leq n_2\}$, $X = \{x_1, x_2, ..., x_{n_2}\}$ and $Y = \{y_1, y_2, ..., y_{n_1}\}$. We define relationships between men in U' and women in W' based on the corresponding vertices in G as follows. Given $r \in \{1, 2\}$ and $p \in \{1, 2, 3\}$, we denote by $w(u_i^r)$ the woman w_j^p where $w_j = w_{i,r}$ and $u_i = u_{j,p}$. Also given $q \in \{1, 2\}$ and $s \in \{1, 2, 3\}$, we denote by $u(w_j^s)$ the man u_i^q where $u_i = u_{j,s}$ and $w_j = w_{i,q}$. The preference lists of the men and women are shown below.

$$
\begin{array}{ll}
u_i^r : y_i \;\; w(u_i^r) \;\; ... & (1 \leq i \leq n_1 \wedge 1 \leq r \leq 2) \\
z_j^s : x_j \;\; w_j^s & (1 \leq j \leq n_2 \wedge 1 \leq s \leq 3) \\
w_j^s : z_j^s \;\; u(w_j^s) \;\; ... & (1 \leq j \leq n_2 \wedge 1 \leq s \leq 3) \\
x_j : z_j^1 \;\; z_j^2 \;\; z_j^3 & (1 \leq j \leq n_2) \\
y_i : u_i^1 \;\; u_i^2 & (1 \leq i \leq n_1)
\end{array}
$$

In the preference list of u_i^r, the symbol **...** means everyone in the set $\{w_a^1, w_a^2, w_a^3, w_b^1, w_b^2, w_b^3\} \backslash \{w(u_i^r)\}$, where $w_a = w_{i,1}$ and $w_b = w_{i,2}$, listed in arbitrary strict order. Also in the preference list of w_j^s, the symbol **...** means everyone in the set $\{u_a^1, u_a^2, u_b^1, u_b^2, u_c^1, u_c^2\} \backslash \{u(w_j^s)\}$, where $u_a = u_{j,1}$, $u_b = u_{j,2}$ and $u_c = u_{j,3}$, listed in arbitrary strict order. We define the set of unacquainted pairs to be

$$U = \{(u_i^1, y_i) : 1 \leq i \leq n_1\} \cup \{(z_j^1, x_j), (z_j^2, x_j) : 1 \leq j \leq n_2\}$$

All other acceptable pairs are acquainted. Let $k' = k + n_1 + 3n_2$. We claim that G has a maximal matching of size $\leq k$ if and only if I has a socially stable matching of size $\leq k'$.

Suppose G has a maximal matching M such that $|M| \leq k$. We construct a matching M' in (I, G') as follows. Let i $(1 \leq i \leq n_1)$ be given. Suppose u_i is matched and $(u_i, w_j) \in M$. Suppose $w_j = w_{i,1}$. Add $(u_i^1, w(u_i^1))$ and (u_i^2, y_i) to M'. Similarly if $w_j = w_{i,2}$ add $(u_i^2, w(u_i^2))$ and (u_i^1, y_i) to M'. Now suppose u_i is unmatched in M. Add (u_i^1, y_i) to

M' (leaving u_i^2 unmatched in M'). Now let j $(1 \leq j \leq n_2)$ be given. Suppose w_j^s is matched in M' for some s $(1 \leq s \leq 3)$. Then w_j^p and w_j^q are unmatched in M' where $\{1,2,3\} = \{p,q,s\}$. Add (z_j^p, w_j^p) to M' and (z_j^q, w_j^q) to M'. Also add (z_j^s, x_j) to M'. Now suppose w_j^s is unmatched in M' for some s $(1 \leq s \leq 3)$. Add (z_j^1, x_j), (z_j^2, w_j^2) and (z_j^3, w_j^3) to M' (leaving w_j^1 unmatched in M'). Then $|M'| \leq |M| + n_1 + 3n_2 \leq k + n_1 + 3n_2$. Thus $|M'| \leq k'$.

We claim that M' is socially stable in I. Suppose not. The pair (u_i^2, y_i) cannot socially block M' as y_i is matched in M' for all i $(1 \leq i \leq n_1)$. The pair (z_j^3, x_j) cannot socially block M' as x_j is matched in M' for all j $(1 \leq j \leq n_2)$. So no z_j^s can be part of any socially blocking pair as every z_j^s is matched in M' for all j $(1 \leq j \leq n_2)$ and s $(1 \leq s \leq 3)$. Also x_j cannot be part of a socially blocking pair as x_j is matched in M' for all j $(1 \leq j \leq n_2)$. Similarly y_i cannot be part of a socially blocking pair as y_i is matched in M' for all i $(1 \leq i \leq n_1)$. Now suppose (u_i^r, w_j^s) socially blocks M'. Then u_i^r is unmatched in M' since u_i^r, if matched in M', never has worse than his 2^{nd} choice. Also w_j^s is unmatched in M' for a similar reason. Thus u_i is unmatched in M and w_j is unmatched in M and $\{u_i, w_j\} \in E(G)$, a contradiction to the maximality of M in G.

Conversely suppose M' is a socially stable matching such that $|M'| \leq k'$. The following facts are easy to establish.

Fact 1. *y_i is matched in M' for all i $(1 \leq i \leq n_1)$ for otherwise (u_i^2, y_i) socially blocks M' in I.*

Fact 2. *x_j is matched in M' for all j $(1 \leq j \leq n_2)$ for otherwise (z_j^3, x_j) socially blocks M' in I.*

Fact 3. *z_j^s is matched in M' for all j $(1 \leq j \leq n_2)$ and s $(1 \leq s \leq 3)$ for otherwise (z_j^s, w_j^s) socially blocks M' in I.*

Suppose $(u_i^r, w_j^s) \in M'$. Add (u_i, w_j) to M. For each i $(1 \leq i \leq n_1)$ there exists at most one r $(1 \leq r \leq 2)$ such that u_i^r is matched to a w_j^s for some j $(1 \leq j \leq n_2)$ and s $(1 \leq s \leq 3)$. by Fact 1. Also for each j $(1 \leq j \leq n_2)$, there exists at most one s $(1 \leq s \leq 3)$ such that w_j^s is matched to a u_i^r for some i $(1 \leq i \leq n_1)$ and r $(1 \leq r \leq 2)$ by Facts 2 and 3. Thus M is a matching in G. Also Facts 1, 2 and 3 imply that $|M| = |M'| - n_1 - 3n_2 \leq k' - n_1 - 3n_2$. Thus $|M| \leq k$. Finally suppose M is not maximal. Then there exists some $(u_i, w_j) \in G$ such that u_i and w_j are both unmatched in M. Thus there exists some u_i^r that is unmatched in M' (by Fact 1) and there exists some w_j^s that is unmatched in M' (by Facts 2 and 3). Since u_i^r finds w_j^s acceptable (u_i^r, w_j^s) socially blocks M' in I, a contradiction to the social stability of M' in I. □

5.5 Further hardness results for SMISS

In this section we present further hardness results relating to SMISS. Once again we consider problems that have previously been investigated in other variants of matching problems involving preferences. Given an instance I of SMISS, we define a man-woman pair m_i, w_j to be a *socially stable pair* if $(m_i, w_j) \in M$ for some socially stable matching M in I. Firstly we investigate the problem SS PAIR-SMSS which is defined as follows. Given an SMSS instance (I, G) and a pair (m, w) in I, determine whether (m_i, w_j) is a socially stable pair in (I, G).

Theorem 5.5.1. SS PAIR-SMSS *is NP-complete.*

Proof. We reduce from COM SMISS, the problem of deciding whether a complete socially stable matching exists in an SMISS instance. Let (I, G) be an SMISS instance with the sets of men and women denoted by $U = \{m_1, m_2, ..., m_n\}$ and $W = \{w_1, w_2, ..., w_n\}$ respectively. We construct an SS PAIR-SMSS instance (I', G') from (I, G) with the set of men and women being $U' = U \cup \{m_0\}$ and $W' = W \cup \{w_0\}$ respectively.

We denote by P_i and Q_j as the preference lists of m_i and w_j in I respectively. In the preference list of m_i, we use the symbol ... to denote the set of women in $\mathcal{W}$ not in P_i. Similarly in the preference list of w_j, we use the symbol ... to denote the set of men in $\mathcal{U}$ not in Q_j. The preference lists of the men and women in I' are shown below.

$$
\begin{aligned}
m_i &: P_i \;\; w_0 \;\; ... && (0 \le i \le n_1)\\
m_0 &: w_1 \;\; w_2 \;\; ... \;\; w_n \;\; w_0 &&\\
w_j &: Q_j \;\; m_0 \;\; ... && (0 \le j \le n_2)\\
w_0 &: m_1 \;\; m_2 \;\; ... \;\; m_n \;\; m_0 &&
\end{aligned}
$$

We construct the social network graph G' with edge-set $E(G') = E(G) \cup \{(m_0, w_j) : 0 \le j \le n\}$. We claim that I admits a complete socially stable matching if and only if (m_0, w_0) is a socially stable pair in I'. Let M be a complete socially stable matching in (I, G). We construct a matching $M' = M \cup \{(m_0, w_0)\}$ in (I', G'). It is easy to see that M' is socially stable in (I', G') as all the men (respectively women) in M have better partners than w_0 (respectively m_0).

Conversely, consider a socially stable matching M' in (I', G') such that $(m_0, w_0) \in M'$. Let $M = M' \backslash \{(m_0, w_0)\}$. We claim that M is a complete socially stable matching in I. If M were not complete and some woman w_j was unmatched in M, then (m_0, w_j) would socially block M', a contradiction to our assumption. Similarly if some pair (m_i, w_j) socially blocks M in I, then (m_i, w_j) would also socially block M' in I' as G is a subgraph of G' , a contradiction to our initial assumption. □

We also consider the problem of determining whether a given man (or woman) is part of at least one socially stable matching. We define the problem MEMBER SMISS as follows: given an SMISS instance (I', G'), determine whether a man m_i belongs to some socially stable matching in (I', G').

Theorem 5.5.2. MEMBER SMISS *is NP-complete.*

Proof. We again reduce from COM SMISS, the problem of deciding whether a complete socially stable matching exists in an SMISS instance. Let (I, G) be an SMISS instance with the sets of men and women denoted by $U = \{m_1, m_2, ..., m_{n_1}\}$ and $W = \{w_1, w_2, ..., w_{n_2}\}$ respectively. We construct an instance of MEMBER SMISS (I', G') from (I, G) with the sets of men and women being $U' = U \cup \{m_0\}$ and $W' = W \cup \{w_0\}$ respectively.

The same meaning for P_i, Q_j and the symbol ... are carried over from the proof of Theorem 5.5.1. We construct the social network graph G' with edge-set $E(G') = E(G) \cup \{(m_i, w_0) : 0 \leq i \leq n_1\}$. The preference lists of the men and women in I' are shown below.

$$
\begin{aligned}
m_i &: P_i \;\; w_0 \;\; ... && (0 \leq i \leq n_1)\\
m_0 &: w_0 &&\\
w_j &: Q_j \;\; ... && (0 \leq j \leq n_2)\\
w_0 &: m_1 \;\; m_2 \;\; ... \;\; m_n \;\; m_0 &&
\end{aligned}
$$

We claim that (I, G) admits a complete socially stable matching if and only if m_0 belongs to some socially stable matching in (I', G'). Let M be a complete socially stable matching in (I, G). We construct a matching $M' = M \cup \{(m_0, w_0)\}$ in (I', G'). Again it is easy to see that M' is socially stable in (I', G').

Conversely, consider a socially stable matching M' in (I', G') such that m_0 is matched in M. Then $(m_0, w_0) \in M'$, let $M = M' \backslash \{(m_0, w_0)\}$. We claim that M is a complete socially stable matching in (I, G). The arguments for the proof are similar to the corresponding argument presented in the proof of Theorem 5.5.1. □

5.6 The roommates problem under social stability

5.6.1 Introduction

So far we have considered social stability in matching problems involving two sets of agents (SMI and HR). In this section we consider social stability in the *Stable Roommates* (SR) context. As defined in Chapter 2, an instance I of SR consists of a set of

agents $R = \{a_1, a_2, ..., a_n\}$. Each agent lists all the other agents in R in strict order of preference. When agents' preference lists are allowed to be incomplete, the problem becomes the *Stable Roomates problem with Incomplete lists* (SRI). A matching is a set of agent pairs such that each agent appears exactly once. A pair $\{a_i, a_j\}$ *blocks* a matching M or forms a *blocking pair* with respect to M if a_i and a_j prefer each other to their partners in M. A matching is *stable* if it admits no blocking pair. A solution to an instance of SR is a stable matching.

In this section, we consider social stability in the SRI context. We investigate the problem of finding a socially stable matching of maximum size and we describe efficient algorithms for some special cases of the problem. We retain the notion of acquainted and unacquainted pairs defined for SMISS in the SRI case. We define an SRISS instance (I, G) as one consisting of an SRI instance I and a social network graph G. The graph G consists of nodes representing the agents in I and edges representing pairs of agents that are acquainted. We denote by U and A the sets of unacquainted and acquainted pairs respectively. A socially blocking pair of a matching M in this context is one that blocks M in the classical sense and is acquainted (i.e., is contained in A). Thus, as is the case with SMISS, only acquainted pairs can socially block a matching. It has already been shown that the problem of finding a socially stable matching or reporting that none exists given an SRI instance is NP-complete [19] [1]. We observe that an instance of SRISS can admit socially stable matchings of various sizes. This can be inferred from an identical observation made for the SMISS case (see Chapter 4) which can be considered as a special case of SRISS. We define MAX SRISS as the problem of finding a maximum socially stable matching or reporting that no socially stable matching exists given an SRISS instance. Given that finding a socially stable matching or reporting that none exists given an SRI instance is NP-complete [19], it follow that MAX SRISS is NP-hard.

Although the general MAX SRISS problem is NP-hard, we can consider special cases that are of practical interest (just as we did in the HRSS and SMISS cases). We observe that for special cases where the sizes of U or A have particular values, the problem of finding a maximum socially stable matching becomes polynomially solvable. If $|U| = 0$ (i.e., $A = \mathcal{A}$) then the problem becomes one of finding a maximum stable matching in the classical SRI problem, which can be solved in polynomial time [50] as all stable matchings are of the same size [38, Theorem 4.5.2]. If $|A| = 0$ (i.e., $U = \mathcal{A}$) then every matching is socially stable and the problem becomes one of finding a maximum matching in a non-bipartite graph (which also can be solved in polynomial time [94]). In the following sections, we consider the cases where U and A are of constant size. We show that under each of these restrictions, MAX SRISS can again be solved in polynomial time.

[1] The authors referred to unacquainted pairs as *free edges*.

5.6.2 MAX SRISS with a constant number of unacquainted pairs

We consider the problem of finding a maximum socially stable matching or reporting that none exists given an SRISS instance when $|U| = k$ for some constant k. Given $S \subseteq U$ we define $I \backslash S$ as an SRISS instance obtained by deleting all the pairs in S from I. We begin by proving an analogue of Lemma 4.6.5 in the SMISS case.

Lemma 5.6.1. *Let (I, G) be an instance of* SRISS. *Let M be a socially stable matching in (I, G). Then there exists a set of unacquainted pairs $U' \subseteq U$ such that M is stable in $I' = I \backslash U'$. Conversely suppose that M is a stable matching in $I' = I \backslash U'$ for some $U' \subseteq U$. Then M is socially stable in (I, G).*

Proof. Suppose M is socially stable in (I, G). Let $U' = U \backslash M$. We claim that M is stable in $I' = I \backslash U'$. Suppose $\{a_i, a_j\}$ blocks M in I'. Then $\{a_i, a_j\} \notin M$. The edges in I' are those in $A \cup (M \cap U)$ but $\{a_i, a_j\} \notin M \cap U$, thus $\{a_i, a_j\} \in A$. Hence $\{a_i, a_j\}$ socially blocks M in (I, G), a contradiction.

Conversely, suppose that M is stable in $I' = I \backslash U'$ for some $U' \subseteq U$. We claim that M is socially stable in (I, G). Suppose that $\{a_i, a_j\}$ socially blocks M in (I, G). Then $\{a_i, a_j\} \in A$ so $\{a_i, a_j\} \notin U$. Thus $\{a_i, a_j\}$ is a pair in I' and so $\{a_i, a_j\}$ blocks M in I', a contradiction. □

By using the technique outlined in Chapter 4 for the HRSS case - considering all subsets $U' \subseteq U$, forming I', finding a stable matching in each such I' (if one exists) and keeping a record of the maximum stable matching found, we obtain a maximum socially stable matching in (I, G). If no stable matching exists in all the instances I' created, then no socially stable matching exists in the instance I. This discussion leads to the following theorem.

Theorem 5.6.2. *Given an instance (I, G) of* SRISS, *we can in $O(2^k m)$ time find a maximum socially stable matching or report that none exists, where m is the number of acceptable pairs and k is the number of unacquainted pairs.*

Proof. The proof is analogous to the one used in Theorem 4.6.6 of Chapter 4. □

5.6.3 MAX SRISS with a constant number of acquainted pairs

We now consider the problem of finding a maximum socially stable matching or reporting that none exists given an SRISS instance when $|A| = k$ for some small integer k. Once again, the approach is an adaption of the one described for this problem in the HRSS case (see Chapter 4). Given the set of acquainted pairs $A = \{e_1, e_2, ..., e_k\}$

we construct a tree T with each node representing pairs $e_i = \{a_{s_i}, a_{t_i}\}$ in A for all i $(1 \leq i \leq k)$ where without loss of generality $s_i < t_i$. The tree T of depth k is constructed with all nodes at depth i labelled e_{i+1} $(i \geq 0)$. There are left and right branches below e_i. Each branch corresponds to a condition placed on a_{s_i} or a_{t_i} with respect to a matching M. The left branch below e_i corresponds to the condition that a_{s_i} is matched in M and prefers his partner to a_{t_i} while the right branch below e_i corresponds to the condition that a_{t_i} is matched in M and is either matched to a_{s_i} or prefers his partner to a_{s_i}. Satisfying at least one of these conditions ensures that $\{a_{s_i}, a_{t_i}\}$ does not socially block M. The tree is constructed in this manner with the nodes at depth $k-1$, labelled e_k, branching in the same way to dummy leaf nodes e_{k+1} (not representing acquainted pairs).

A path P from the root node e_1 to a leaf node e_{k+1} will visit all pairs in A exactly once. Every left branch in P gives a *left condition* and every right branch gives a *right condition*. Let R' be the set of agents involved in left and right conditions in P. It is possible that an agent could be involved in more than one condition, and indeed in both left and right conditions. Given a matching M, enforcing all the conditions along P can be achieved by first deleting all pairs from the instance I that could potentially violate these conditions. So if some left condition along P states that agent a_{s_i} must be matched in M to an agent he prefers to a_{t_i} then a_{s_i}'s preference list is truncated starting with a_{t_i}. If some right condition states that an agent a_{t_i} must be matched in M to an agent no worse than a_{s_i} then a_{t_i}'s preference list is truncated starting from the agent immediately following a_{s_i}. After performing these truncations based on the conditions along P, a new SRISS instance I' is obtained.

Lemma 5.6.3. *If M is a matching in I' that is computed at the leaf node of a path P and all agents in R' are matched in M then M is a socially stable matching in (I, G).*

Proof. The proof is analogous to the one used in Lemma 4.6.7 of Chapter 4. □

With I' obtained due the truncations carried out by satisfying conditions along a path P from the root node to a leaf node, we then seek to obtain a matching in which all the agents in R' are matched. We construct a graph G' consisting of nodes representing the agents in I' and weighted edges representing the acceptable pairs in I'. We mark all the nodes representing the agents in R' as red nodes with the remaining nodes uncoloured. We place weights on the edges as follows: (i) an edge between a red node and an uncoloured node is given a weight of 1; (ii) an edge between two red nodes is given weight of 2; (iii) an edge between two uncoloured nodes is given a weight of 0. We then find a maximum weight matching M' in the resulting weighted graph G'. Let $wt(M')$ denote the weight of a matching M' in G'. Then

$$wt(M') = |\{a_i \in R' : a_i \text{ is matched in } M'\}| \leq |R'|$$

Moreover $wt(M') = |R'|$ if and only if every agent in R' is matched in M'. For such a matching M' in G', a maximum cardinality matching M'' can be obtained in G' by continuously augmenting M' until no augmenting path can be found. Since any node already matched in M' will remain matched in M'' it follows that all the agents in R' will be matched in M'' and such a matching, by Lemma 5.6.3, will be socially stable in (I, G). If however, $wt(M') < |R'|$ then some agent in R' remains unmatched in any maximum matching in G' thus introducing the possibility of a socially blocking pair of M' in (I, G). In this case, P is ruled as infeasible and another path is considered, otherwise P is called feasible.

There are 2^k paths from the root node to leaf nodes in the tree T. The following proposition is important to our result.

Lemma 5.6.4. *If (I, G) admits a socially stable matching, there must exist at least one feasible path in T.*

Proof. The proof is analogous to the one used in Lemma 4.6.8 of Chapter 4. □

To generate a maximum socially stable matching M if one exists in an instance (I, G) of SRISS, all 2^k paths through T from the root node to leaf nodes are considered with a record kept of the largest matching M (satisfying the constraints of Proposition 5.6.3) computed at the leaf node of each feasible path. M is then the desired matching as the following proposition shows

Proposition 5.6.5. *If M is a matching obtained from the process described above, M is a maximum socially stable matching in (I, G).*

Proof. The proof is analogous to the one used in Proposition 6.9 of Chapter 4. □

The above proposition leads to the following main result of this subsection.

Theorem 5.6.6. *Given an instance (I, G) of* SRISS *where the set A of acquainted pairs satisfies $|A| = k$, a maximum socially stable matching can be generated in $O(2^k \sqrt{n} m \log_n n^2/m)$ time where n is the number of agents and m is the number of acceptable pairs.*

Proof. The time complexity is dominated by the problem of finding a maximum weight matching in a general graph. This can be solved in $O(\sqrt{n} m \log_n n^2/m)$ time [47]. This needs to be done 2^k times. □

Following the results in Theorems 5.6.2 and 5.6.6, we conclude this section with the theorem below showing the existence of FPT algorithms for MAX SRISS under two different parameterisations.

Corollary 5.6.7. MAX SRISS *is in FPT with parameter k where k is the number of unacquainted pairs.*

Corollary 5.6.8. MAX SRISS *is in FPT with parameter k where k is the number of acquainted pairs.*

5.6.4 2-MAX SRISS

In the MAX SMISS problem, we have shown that when the lengths of the residents' preference lists are at most 2, the problem can be solved in polynomial time. The case of short preference list is significant as some applications of matching problems tend to limit the lengths of the preference lists for one or both sets of agents. Another motivation for considering matching problems with bounded-length preference lists is the observation that the problems tend to switch from polynomial solvability to NP-hardness at some preference list length. For example we know that $(2, \infty)$-MAX SMISS in solvable in polynomial time but $(3, \infty)$-MAX SMISS is NP-hard (see Theorem 4.4.2). In this subsection we consider similar restrictions in the roommates case. We denote by K-MAX SRISS the problem of finding a maximum socially stable matching or reporting that no socially stable matching exists in an SRISS instance in which the preference lists are of length at most k. We present a polynomial-time algorithm for 2-MAX SRISS in the knowledge that 3-MAX SRISS is NP-hard (a consequence of Theorem 4.4.2).

Before exploring the algorithm we identify three types of socially blocking pairs that may exist in the 2-MAX SRISS context. Consider two agents a_i and a_j in a 2-MAX SRISS instance (I, G) such that $\{a_i, a_j\} \in A$. For a given matching M in (I, G), if $\{a_i, a_j\} \notin M$, then $\{a_i, a_j\}$ can form a socially blocking pair of one of three types with respect to M in I. (i) In a type-1 socially blocking pair, a_j is first on a_i's list and a_i is first on a_j's list. (ii) In a type-2 socially blocking pair a_i is unmatched in M and a_j prefers a_i to $M(a_j)$. (iii) In a type-3 socially blocking pair both a_i and a_j are unmatched in M.

The first phase of the algorithm involves identifying agent pairs that must be matched in any socially stable matching. Specifically if two agents a_i and a_j appear first on each others' preference lists and form an acquainted pair then they must be matched to each other in any socially stable matching. We simply add them to the final matching produced and remove them from the instance. This process continues until no such pair exists.

In the second phase of the algorithm we generate a graph G' from the resulting instance I' (where the vertices and edges represent agents and acceptable pairs respectively) and find a maximum matching M' in G'. Connected components of G' are paths and cycles.

Finally in the third phase, we remove certain socially blocking pairs which may still exist in M'. At this stage only type-2 blocking pairs can potentially be present in M'. We attempt to satisfy such blocking pairs by assigning the agents involved to each other. For example if $\{a_i, a_j\}$ forms a type-2 blocking pair with respect to M' such that a_i is unmatched in M', if $a_k = M'(a_j)$ we remove $\{a_j, a_k\}$ from M' and add $\{a_i, a_k\}$ to M'. We then mark a_k as *visited*. We continue to satisfy these blocking pairs in a similar way by assigning the agents involved to each other. If an agent previously marked as *visited* is involved in a type-2 blocking pair again, we call the sequence of blocking pairs that were satisfied between the previous and current occurrence of such an agent an *odd-length cycle* of blocking pairs. In such a cycle a type-2 blocking pair will always exist with respect to any maximal matching in the cycle. An example of such a cycle is shown in Figure 5.6 in which all pairs are acquainted. If any two of the three agents are matched to each other, the resulting matching would admit a type-2 blocking pair. If such a cycle is found, the instance is reported as unsolvable. Otherwise all type-2 blocking pairs with respect to M' will be satisfied. The algorithm finally reports the union of M' and the set of pairs removed in Phase 1 of the algorithm. Algorithm 5.1 describes the full 2-`max sriss-alg` algorithm.

$$a_1 : a_2 \quad a_3$$
$$a_2 : a_3 \quad a_1$$
$$a_3 : a_1 \quad a_2$$

Figure 5.6: A cycle of type-2 blocking pairs

We now show that the algorithm produces a maximum socially stable matching should one exist. Firstly, it is easy to see that phases 1 and 2 terminate. As for phase 3, every time a blocking pair is satisfied both agents involved become better off with one previously matched agent becoming unmatched. This overall improvement can only occur a finite number of times before no such blocking pair can be found or an odd-length cycle of type-2 blocking pairs is found.

Lemma 5.6.9. *If a matching is returned by* 2-`max sriss-alg` *it is socially stable in* (I, G).

Proof. Suppose the matching M produced by 2-`max sriss-alg` is not socially stable in (I, G). Then some pair $\{a_i, a_j\} \in A$ must form a type-1, 2 or 3 socially blocking pair with respect to M in (I, G).

Algorithm 5.1 2-max sriss-alg

```
1: /* Phase 1 */
2: M := {};
3: while some pair {a_i, a_j} ∈ A exists where both appear first on each other's list do
4:    M = M ∪ {{a_i, a_j}};
5:    remove the pair {a_i, a_j} from the instance;
6: /* Phase 2 */
7: construct G';
8: M' := maximum matching in G';
9: /* Phase 3 */
10: while some type-2 blocking pair {a_i, a_j} ∈ A exists such that a_i is matched in M' do
11:    if a_j is marked as visited then
12:       return with status: unsolvable;
13:    mark M'(a_i) as visited;
14:    M' := M'\{{a_i, M(a_i)}};
15:    M' := M' ∪ {{a_i, a_j}};
16: return M ∪ M';
```

Case (i): $\{a_i, a_j\}$ is a type-1 blocking pair. This case satisfies the loop condition in Phase 1 and thus should never arise once Phase 1 has terminated.

Case (ii): $\{a_i, a_j\}$ is a type-2 blocking pair where a_i is unmatched in M and a_j prefers a_i to $M(a_j)$. This case satisfies the loop condition in Phase 3 and thus should never arise once Phase 3 has terminated.

Case (iii): $\{a_i, a_j\}$ is a type-3 blocking pair where a_i and a_j are unmatched in M. Such a blocking pair could not exist after phase 2 of the algorithm. Thus it must have been created due to some step in phase 3. Phase 3 iterates when an even-length path or an odd-length cycle exists. In each of these components only one vertex is unmatched in M'. Furthermore any switch caused by Phase 3 still leaves only one vertex unmatched, thus a contradiction. □

Since (i) by Lemma 5.6.9, the matching produced is a socially stable matching, (ii) the pairs removed in Phase 1 will belong to all socially stable matchings, (iii) M' is a maximum matching in G' and (iv) Phase 3 never reduces the size of the matching, it follows that the matching produced by the algorithm is a maximum socially stable matching in (I, G).

The complexity of the algorithm is dominated by Phase 2. The complexity of the algorithm for finding a maximum matching in G' is $O(m)$ where m is the number of acceptable pairs. This is possible as G' consists of only cycles and paths which can be iterated over in linear time. We have thus proved the following theorem.

Theorem 5.6.10. *Given an instance* (I, G) *of* 2-MAX SRISS, *Algorithm* 2-max sriss-alg

generates a maximum socially stable matching or reports that no socially stable matching exists in $O(m)$ time, where m is the number of acceptable pairs.

5.7 Conclusion

We have presented a wide range of algorithmic results relating to finding various types of socially stable matchings in both two-sided and one-sided matching problems. Although a majority of the results indicate NP-hardness, we still consider it worthwhile adopting social stability as a solution criterion in practice. Efficient algorithms for various special cases have been shown. Moreover the empirical evaluations presented in Chapter 4 demonstrate the practicality of the approximation algorithms presented.

Chapter 6

Profile-based optimal matchings in the Student/Project Allocation problem

6.1 Introduction

In Section 2.6 we introduced the *Student/Project Allocation problem* (SPA) and gave some motivation for studying SPA and its variants. Although the SPA problem finds its main application in the academic context where we seek to match students to individual or group projects, it can also be used to solve other allocation problems having the same structure or characteristics such as the assignment of employees to posts in a company where available posts are offered by various departments. We also presented a review of the current literature with respect to SPA in Section 2.6. A considerable amount of the prior work in this area is based on either efficient algorithms for finding stable matchings in SPA problems involving two-sided preferences, or integer programming techniques for SPA variants involving other optimality criteria. In the case of one-sided preference lists (which we have shown to be of practical relevance), the notion of stability becomes irrelevant. Other optimality criteria based on the profile of a matching, can be considered

We highlight some major approaches for solving these profile-based optimal SPA problems and highlight their drawbacks, as follows:

1. Using algorithms for the well-known *Capacitated House Allocation problem with Ties* (CHAT) [54, 93, 110, 48]. While efficient algorithms for this problem exist, such algorithms require restricting SPA to cases where lecturer upper or lower

quotas are not considered. In scenarios where load balancing and load capping are important, this approach becomes infeasible.

2. Using algorithms for the *Minimum Cost Maximum Flow problem* (MCMF) [5, 82, 116]. While these approaches find profile-based optimal matchings, they suffer from scalability issues due to exponentially-large edge weights that arise due to their cost functions.

3. Using integer programming [10, 113, 109, 70] and constraint programming [25, 112] models to solve SPA problems. While these techniques are very flexible and may cater for a wide variety of applications, they have theoretically exponential time complexities. This, as we demonstrate later, may translate to unscalable solutions in practice.

In this chapter, we begin to address some of these drawbacks. We model SPA as a network flow problem and describe a modified augmenting path algorithm for finding a maximum flow which can then be transformed to a profile-based optimal SPA matching. This approach avoids the use of large edge weights and introduces greater flexibility by allowing side constraints such as lecturer lower quotas to be added to the model. In Section 6.2 we formally define the SPA model. In Section 6.3 we present an efficient algorithm for finding a greedy maximum matching given a SPA instance and prove its correctness. The algorithm takes lecturer upper quotas into consideration and can be slightly modified to consider lecturer lower quotas. In Section 6.4 we show how this algorithm can be modified in order to find a generous maximum matching. Section 6.5 describes two IP models for SPA which we use for the correctness testing of our implemented algorithms. In Section 6.6 we present results from an empirical evaluation of the algorithms and IP models presented. We conclude the chapter in Section 6.7 by presenting some relevant open problems.

6.2 Preliminary definitions

An instance I of the SPA problem consists of a set $\mathcal{S}$ of students, a set $\mathcal{P}$ of projects and a set $\mathcal{L}$ of lecturers. Each student s_i ranks a set $A_i \subseteq \mathcal{P}$ of projects that she considers acceptable in order of preference. This *preference list* of projects may contain ties. Each project $p_j \in \mathcal{P}$ has an upper quota c_j indicating the maximum number of students that can be assigned to it. Each lecturer $l_k \in \mathcal{L}$ offers a set of projects $P_k \subseteq \mathcal{P}$ and has an upper quota d_k^+ indicating the maximum number of students that can be assigned to l_k. Unless explicitly mentioned, we assume that all lecturer lower quotas are equal to

0. The sets $\{P_1, \ldots, P_k\}$ partition $\mathcal{P}$. If project $p_j \in P_k$, then we denote the lecturer assigned to p_j as $l_k = l(p_j)$.

An *assignment* M in I is a subset of $\mathcal{S} \times \mathcal{P}$ such that:

1. Student-project pair $(s_i, p_j) \in M$ implies $p_j \in A_i$.
2. For each student $s_i \in \mathcal{S}$, $|\{(s_i, p_j) \in M : p_j \in A_i\}| \leq 1$.

If $(s_i, p_j) \in M$ we denote $M(s_i) = p_j$. For a project p_j, $M(p_j)$ is the set of students assigned to p_j in M. Also if $(s_i, p_j) \in M$ and $p_j \in P_k$ we say student s_i is assigned to project p_j and to lecturer l_k in M. We denote the set of students assigned to a lecturer l_k as $M(l_k)$. A *matching* in this problem is an assignment M that satisfies the capacity constraints of the projects and lecturers. That is, $|M(p_j)| \leq c_j$ for all projects $p_j \in \mathcal{P}$ and $|M(l_k)| \leq d_k^+$ for all lecturers $l_k \in \mathcal{L}$.

Given a student s_i and a project $p_j \in A_i$, we define $rank(s_i, p_j)$ as 1 + the number of projects that s_i prefers to p_j. Let R be the maximum rank of a project in any student's preference list. We define the *profile* $\rho(M)$ of a matching M in I as an R-tuple $(x_1, x_2, ..., x_R)$ where for each r $(1 \leq r \leq R)$, x_r is the number of students s_i assigned in M to a project p_j such that $rank(s_i, p_j) = r$. Let $\alpha = (x_1, x_2, ..., x_R)$ and $\sigma = (y_1, y_2, ..., y_R)$ be any two profiles. We define the *empty profile* $O_R = (o_1, o_2, ..., o_R)$ where $o_r = 0$ for all r $(1 \leq r \leq R)$. We also define the *negative infinity profile* $B_R^- = (b_1, b_2, ..., b_R)$ where $b_r = -\infty$ $(1 \leq r \leq R)$ and the *positive infinity profile* $B_R^+ = (b_1, b_2, ..., b_R)$ where $b_r = \infty$ $(1 \leq r \leq R)$. We define the sum of two profiles α and σ as $\alpha + \sigma = (x_1 + y_1, x_2 + y_2, ..., x_R + y_R)$. Given any q $(1 \leq q \leq R)$, we define $\alpha + q = (x_1, ..., x_{q-1}, x_q + 1, x_{q+1}, ..., x_R)$. We define $\alpha - q$ in a similar way.

We define the total order $\succ_L$ on profiles as follows. We say α *left dominates* σ, denoted by $\alpha \succ_L \sigma$ if there exists some r $(1 \leq r \leq R)$ such that $x_{r'} = y_{r'}$ for $1 \leq r' < r$ and $x_r > y_r$. We define *weak left domination* as follows. We say $\alpha \succeq_L \sigma$ if $\alpha = \sigma$ or $\alpha \succ_L \sigma$. We may also define an alternative total order $\prec_R$ on profiles as follows. We say α *right dominates* σ $(\alpha \prec_R \sigma)$ if there exists some r $(1 \leq r \leq R)$ such that $x_{r'} = y_{r'}$ for $r < r' \leq R$ and $x_r < y_r$. We also define *weak right domination* as follows. We say $\alpha \preceq_R \sigma$ if $\alpha = \sigma$ or $\alpha \prec_R \sigma$. Intuitively α *left dominates* σ if α is lexicographically greater than σ. Also α *right dominates* σ if σ is lexicographically less than α when considered in the reverse order.

The SPA problem can be modelled as a network flow problem. Given a SPA instance I, we construct a flow network $N(I) = \langle G, c \rangle$ where $G = (V, E)$ is a directed graph and c is a non-negative capacity function $c : E \to \mathbb{R}^+$ defining the maximum flow allowed through each edge in E. The vertices in G correspond to the agents (students,

projects and lecturers) in the instance. A source vertex v_s and a sink vertex v_t are also included. The edges in G correspond to potential assignments between the agents involved. The capacities of these edges are set in order to reflect the upper bounds placed on these assignments. For example edges that represent potential assignments between a student s_i and a project p_j will have a capacity of 1 as s_i can only be assigned to a p_j once. The construction of $N(I)$ is such that there is a one-to-one correspondence between the set of feasible flows in $N(I)$ and feasible matchings in I. The network is constructed as follows. Let $V = \{v_s, v_t\} \cup \mathcal{S} \cup \mathcal{P} \cup \mathcal{L}$ and $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where $E_1 = \{(v_s, s_i) : s_i \in \mathcal{S}\}$, $E_2 = \{(s_i, p_j) : s_i \in \mathcal{S}, p_j \in A_i\}$, $E_3 = \{(p_j, l_k) : p_j \in \mathcal{P}, l_k = l(p_j)\}$ and $E_4 = \{(l_k, v_t) : l_k \in \mathcal{L}\}$. We set the capacities as follows: $c(v_s, s_i) = 1$ for all $(v_s, s_i) \in E_1$, $c(s_i, p_j) = 1$ for all $(s_i, p_j) \in E_2$, $c(p_j, l_k) = c_j$ for all $(p_j, l_k) \in E_3$ and $c(l_k, v_t) = d_k^+$ for all $(l_k, v_t) \in E_4$.

We call a path P' from v_s to some project p_j a *partial augmenting path* if P' can be extended adding the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ to form an augmenting path with respect to flow f. Given a partial augmenting path P' from v_s to p_j the *profile* of P', denoted $\rho(P')$, shows the effect P' would have on the profile of a matching $M(f)$ obtained from f if P' were augmented along f.

$$\rho(P') = O_R + \sum\{rank(s_i, p_j) : (s_i, p_j) \in P' \wedge f(s_i, p_j) = 0\} - \sum\{rank(s_i, p_j) : (p_j, s_i) \in P' \wedge f(s_i, p_j) = 1\}$$

where additions are done with respect to the $+$ and $-$ operations on profiles. Unlike the profile of a matching, the profile of an augmenting path may contain negative values. Also if P' can be extended to a full augmenting path P with respect to flow f by adding the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ where v_s and p_j are the endpoints of P', then we define the profile of P, denoted by $\rho(P)$, to be $\rho(P) = \rho(P')$. Multiple partial augmenting paths may exist from v_s to p_j, thus we define the *maximum profile of a partial augmenting path* from v_s to p_j with respect to $\succ_L$, denoted $\Phi(p_j)$, as follows:

$$\Phi(p_j) = \max_{\succ_L}\{\rho(P') : P' \text{ is a partial augmenting path from } v_s \text{ to } p_j\}.$$

An augmenting path P is called a *maximum profile augmenting path* if

$$\rho(P) = \max_{\succ_L}\{\Phi(p_j) : p_j \in \mathcal{P}\}.$$

Let f be an integral flow in N. We define the matching $M(f)$ in I induced by f as follows: $M(f) = \{(s_i, p_j) : f(s_i, p_j) = 1\}$. Clearly by construction of N, $M(f)$ is a matching in I, such that $|M(f)| = |f|$. If P is an augmenting path with respect to f then $\rho(M') = \rho(M) + \rho(P)$ where $M = M(f), M' = M(f')$ and f' is the flow obtained

students' preferences:	lecturers' offerings:
$s_1 : p_1 \quad p_2 \quad p_3$	$l_1 : \{p_1, p_2\}$
$s_2 : p_1$	$l_2 : \{p_3\}$
$s_3 : p_2 \quad p_3$	project capacities: $c_1 = 1, c_2 = 1, c_3 = 1$
	lecturer capacities: $d_1 = 2, d_2 = 1$

Figure 6.1: A SPA instance I

by augmenting f along P. Also given a matching M in I, we define a flow $f(M)$ in N corresponding to M as follows:

$\forall\ (v_s, s_i) \in E_1,\ f(v_s, s_i) = 1$ if s_i is matched in M and $f(v_s, s_i) = 0$ otherwise.
$\forall\ (s_i, p_j) \in E_2,\ f(s_i, p_j) = 1$ if $(s_i, p_j) \in M$ and $f(s_i, p_j) = 0$ otherwise.
$\forall\ (p_j, l_k) \in E_3,\ f(p_j, l_k) = c'_j$ where $c'_j = |M(p_j)|$
$\forall\ (l_k, v_t) \in E_4,\ f(l_k, v_t) = d'_k$ where $d'_k = |M(l_k)|$

We define a student s_i to be *exposed* if $f(v_s, s_i) = 0$ meaning that there is no flow through s_i. Similarly we define a project p_j to be *exposed* if $f(p_j, l_k) < c_j$ and $f(l_k, v_t) < d_k^+$ where $l_k = l(p_j)$.

Let M be a matching of size k in I. We say that M is a *greedy k-matching* if there is no other matching M' such that $|M'| = k$ and $\rho(M') \succ_L \rho(M)$. If k is the size of a maximum cardinality matching in I, we call M a *greedy maximum matching* in I. Also we say that M is a *generous k-matching* if there is no other matching M' such that $|M'| = k$ and $\rho(M') \prec_R \rho(M)$. If k is the size of a maximum cardinality matching in I, we call M a *generous maximum matching* in I.

Figure 6.1 shows a sample SPA instance with greedy and generous maximum matchings $M_1 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\}$ and $M_2 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ respectively.

6.3 Greedy maximum matchings in SPA

In this section we present the algorithm `Greedy-max-spa` for finding a greedy maximum matching given a SPA instance. The algorithm is based on the general Ford-Fulkerson algorithm for finding a maximum flow in a network [27]. We obtain maximum profile augmenting paths by adopting techniques used in the bipartite matching approach for finding a greedy maximum matching in HA [54] and CHA [110].

The `Greedy-max-spa` algorithm shown in Algorithm 6.1 takes in a SPA instance I as input and returns a greedy maximum matching M in I. A flow network $N(I) = \langle G, c \rangle$ is constructed as described in Section 6.2. Given a flow f in $N(I)$ that yields a greedy k-matching $M(f)$ in I, if k is not the size of a maximum flow in $N(I)$, we seek to find

a maximum profile augmenting path P with respect to f in $N(I)$ such that the new flow f' obtained by augmenting f along P yields a greedy $(k+1)$-matching $M(f')$ in I. Lemmas 6.3.1 and 6.3.2 show the correctness of this approach. In Lemma 6.3.1 we show that if k is smaller than the size of a maximum flow in $N(I)$ then such a path P is bound to exist. In Lemma 6.3.2 we show that P is a maximum profile augmenting path with respect to f. Next we give an overview of the `Greedy-max-spa` algorithm and proove in Lemma 6.3.3 that `Greedy-max-spa` produces P. We conclude the section with Theorem 6.3.4 stating the main result that finding a greedy maximum matching given a SPA instance can be done in polynomial time.

Lemma 6.3.1. *Let I be an instance of* SPA *and let η denote the size of a maximum matching in I. Let k $(1 \leq k < \eta)$ be given and suppose that M_k is a greedy k-matching in I. Let $N = N(I)$ and $f = f(M_k)$. Then there exists an augmenting path P with respect to f in N such that if f' is the result of augmenting f along P then $M_{k+1} = M(f')$ is a greedy $(k+1)$-matching in I.*

Proof. In the proof we examine the symmetric difference between M_k and M_{k+1}. We hope to show that $P = M_k \oplus M_{k+1}$ is an augmenting path. To achieve this we consider the any greedy $(k+1)$-matching M'_{k+1} and analyse the connected components in $M_k \oplus M'_{k+1}$. We show that symmetric difference between certain groups of these connected components and M'_{k+1} will yield other greedy $(k+1)$-matchings. By sucessfully applying this strategy we eventually discover a greedy $(k+1)$-matching such that $P = M_k \oplus M_{k+1}$ is an augmenting path.

Let $I' = C(I)$ be a new instance of SPA obtained from I as follows. Firstly we add all students in I to I'. Next, for every project $p_j \in \mathcal{P}$, we add c_j clones $p_j^1, p_j^2, ..., p_j^{c_j}$ to I' each of capacity 1. We then add all lecturers in I to I'. If $p_j \in A_i$ in I, we add (s_i, p_j^r) to I' for all r $(1 \leq r \leq c_j)$. If $p_j \in P_k$ is in I, we add (p_j^r, l_k) to I' for all r $(1 \leq r \leq c_j)$. Also if $rank(s_i, p_j) = t$, we set $rank(s_i, p_j^r) = t$ for all r $(1 \leq r \leq c_j)$. Let G' be the underlying graph in I' involving only the student and project clones. With respect to the matching $M_k = M(f)$, we construct a cloned matching $C(M_k)$ in I' as follows. If project p_j is assigned x_j students $s_{q,1}, s_{q,2}, ..., s_{q,x_j}$ in M_k we add $(s_{q,r}, p_j^r)$ to $C(M_k)$ for all $1 \leq r \leq x_j$. Hence $C(M_k)$ is a greedy k-matching in I'.

Let M'_{k+1} be a greedy $(k+1)$-matching in I (this exists because $k < \eta$). Then $C(M'_{k+1})$ is a greedy $(k+1)$-matching in I'. Let $X = C(M_k) \oplus C(M'_{k+1})$ be the symmetric difference between $C(M_k)$ and $C(M'_{k+1})$. Then each connected component of X is either (i) an alternating cycle, (ii) an even-length alternating path or (iii) an odd-length alternating path in G' (with no restrictions on which matching the end edges belong to). The aim is to show that, by eliminating a subset of X, we are left with a set of connected components which can be transformed into a single augmenting path with

respect to $f(C(M_k))$ in $N(I')$ and subsequently a single augmenting path with respect to $f(M_k)$ in $N(I)$.

Eliminating connected components of X: Suppose $D \subseteq X$ is a type (i) connected component of X or a type (ii) connected component of X whose end vertices are students (we may call this a type (ii)(a) component). Suppose also that $\rho(D \cap C(M'_{k+1})) \succ_L \rho(D \cap C(M_k))$. A new matching $C(M'_k)$ in G' of cardinality k can be created from $C(M_k)$ by replacing all the $C(M_k)$-edges in D with the $C(M'_{k+1})$-edges in D (i.e. by augmenting $C(M_k)$ along D). Since the upper quota constraints of the lecturers involved are not violated after creating $C(M'_k)$ from $C(M_k)$, it follows that $C(M'_k)$ is also a valid SPA matching in I'. Moreover $\rho(C(M'_k)) \succ_L \rho(C(M_k))$ which is a contradiction to the fact that $C(M_k)$ is a greedy k-matching in I'. A similar contradiction (to the fact that $C(M'_{k+1})$ is a greedy $(k+1)$- matching in I') exists if we assume $\rho(D \cap C(M_k)) \succ_L \rho(D \cap C(M'_{k+1}))$. Thus $\rho(D \cap C(M'_{k+1})) = \rho(D \cap C(M_k))$.

Form the argument above, no type (i) or type (ii)(a) connected component of X contributes to a change in the size or profile as we augment from $C(M_k)$ to $C(M'_{k+1})$ or vice versa. In fact, this is true for any even-length connected component of X which does not cause lecturer upper quota constraints to be violated as we augment from $C(M_k)$ to $C(M'_{k+1})$ or vice versa. The claim can be extended to certain groups of connected components which, when considered together, (i) have equal numbers of $C(M_k)$ and $C(M'_{k+1})$ edges and (ii) do not cause lecturer upper quota constraints to be violated as we augment from $C(M_k)$ to $C(M'_{k+1})$ or vice versa. In all these cases, it is possible to *eliminate* such components (or groups of components) from consideration. Using the above reasoning, we begin by eliminating all type (i) and type (ii)(a) connected components of X.

Let $\mathcal{D}$ be the union of all the edges in type (i) and type (ii)(a) connected components of X. Let $X' = X \backslash \mathcal{D}$ and let $C(M''_{k+1})$ be some greedy $(k+1)$-matching in I' which can be constructed by augmenting $C(M'_{k+1})$ along $\mathcal{D}$. Then it follows that $X' = C(M_k) \oplus C(M''_{k+1})$. Thus X' contains (1) even-length alternating paths whose end vertices are project clones (we call these type (ii)(b) paths), (2) odd-length alternating paths whose end edges are in $C(M_k)$ (we call these type (iii)(a) paths) and (3) odd-length alternating paths whose end edges are in $C(M''_{k+1})$ (we call these type (iii)(b) paths). Although these alternating paths are vertex disjoint, there are special cases where two alternating paths in X' may be *joined together* by pairing their end project clone vertices.

Joining alternating paths: Consider some lecturer l_q and project $p_j \in P_q$. We

extend the notation $l(p_j)$ to include all clones of p_j (i.e. $l(p_j^r) = l_q$ for all r $(1 \leq r \leq c_j)$). Let

$$X_q = \{(s_i, p_j^r) \in C(M_k) : l_q = l(p_j^r) \wedge p_j^r \text{ is unmatched in } C(M''_{k+1})\}$$

and let $x_q = |X_q|$. Thus X_q is the set of end edges incident to project clones belonging to a subset of the type (ii)(b) and type (iii)(a) paths in X'. Let

$$Y_q = \{(s_i, p_j^r) \in C(M''_{k+1}) : l_q = l(p_j^r) \wedge p_j^r \text{ is unmatched in } C(M_k)\}$$

and let $y_q = |Y_q|$. Thus Y_q is the set of end edges incident to project clones belonging to a subset of the type (ii)(b) and type (iii)(b) paths in X'. Also let

$$Z_q = \{p_j^r : l_q = l(p_j^r) \wedge p_j^r \text{ is matched in } C(M''_{k+1}) \wedge p_j^r \text{ is matched in } C(M_k)\}$$

and let $z_q = |Z_q|$. Thus $d_q = v_q + x_q + z_q$ and $d_q = v'_q + y_q + z_q$ where v_q and v'_q are the number of unassigned positions that l_q has in $C(M_k)$ and $C(M''_{k+1})$ respectively.

Note that $v_q \geq y_q$ if and only if $v'_q \geq x_q$. Since $v_q \geq y_q$, all the paths with end edges in Y_q can be considered as valid alternating paths in $C(M_k)$ (i.e. if they are used to augment $C(M_k)$, l_q's upper quota will not be violated in the resulting matching). Since $v'_q \geq x_q$, all the paths with end edges in X_q can be considered as valid alternating paths in $C(M''_{k+1})$ (i.e. if they are used to augment $C(M''_{k+1})$, l_q's upper quota will not be violated in the resulting matching).

On the other hand, assume $y_q > v_q$. Then $x_q > v'_q$. Let $Y'_q \subseteq Y_q$ be an arbitrary subset of Y_q of size v_q and let $X'_q \subseteq X_q$ be an arbitrary subset of X_q of size v'_q. Thus all paths with end edges in X'_q and Y'_q can be considered as valid alternating paths in $C(M''_{k+1})$ and $C(M_k)$ respectively. Also $|Y_q \backslash Y'_q| = y_q - v_q = |X_q \backslash X'_q| = x_q - v'_q$. We can thus form a $1-1$ correspondence between the edges in $|Y_q \backslash Y'_q|$ and those in $|X_q \backslash X'_q|$. Let $(s_i, p_j^r) \in Y_q \backslash Y'_q$ and $(s_{i'}, p_{j'}^{r'}) \in X_q \backslash X'_q$ be the end edges of two alternating paths in X'. The paths can be joined together by *pairing* the clones of both end projects thus forming a *project pair* $(p_j^r, p_{j'}^{r'})$ at l_q. These project pairs can be formed from all edges in $Y_q \backslash Y'_q$ and $X_q \backslash X'_q$.

In the cases where project pairs are formed, the resulting path (which we call a *compound path*) may be regarded as a single path along which $C(M_k)$ or $C(M''_{k+1})$ may be augmented. In some cases, the two projects being paired may be end vertices of a single (or compound) alternating path. Thus pairing them together will form a cycle. Since the cycle is of even length and the lecturer's upper quota will not be violated if it is used to augment $C(M_k)$ or $C(M''_{k+1})$ it can be eliminated right away. For each lecturer $l_q \in \mathcal{L}$, once the pairings between alternating paths in $Y_q \backslash Y'_q$ and $X_q \backslash X'_q$ have been carried out (where applicable) and any formed cycles have been eliminated, we

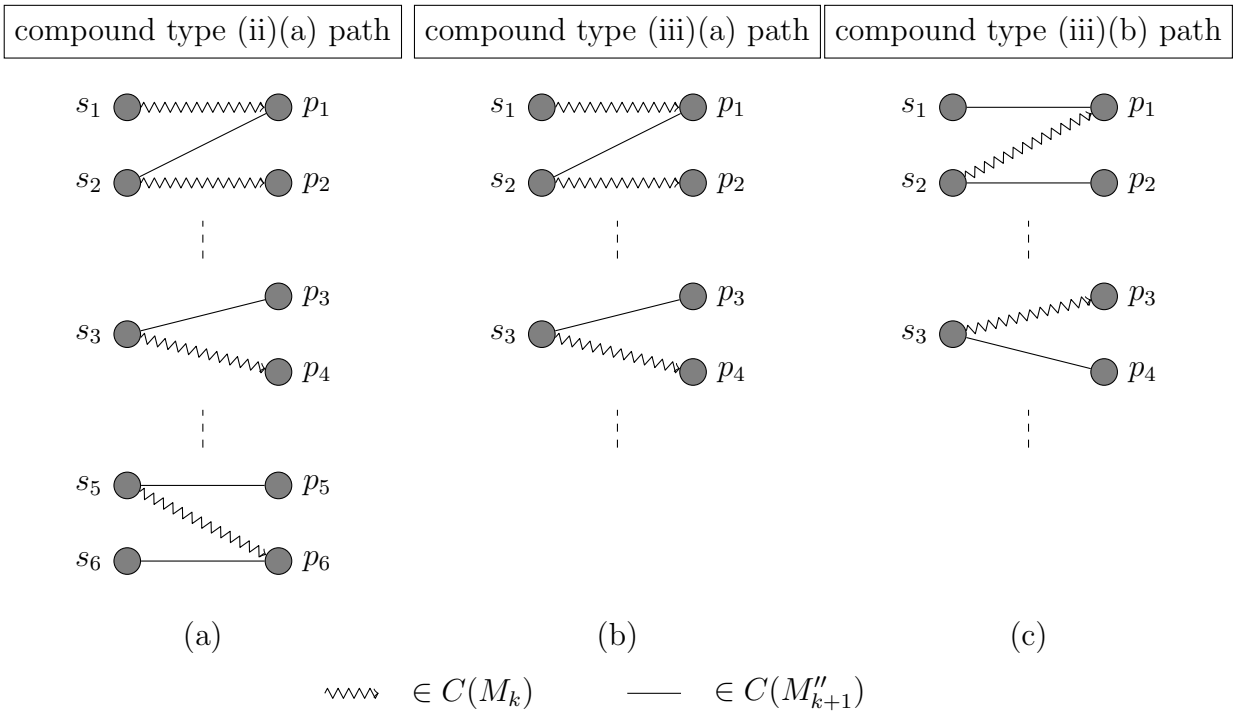

Figure 6.2: Some types of compound path in X'

are left with a set of single or compound alternating paths of the following types (for simplicity we call all remaining alternating paths *compound paths* even though they may consist of only one path).

1. A *compound type (ii)(a) path* - a compound path with an even number of edges with both end vertices being students. This path will contain a type (iii)(a) path at one end, and a type (iii)(b) path at the other end with zero or more type (ii)(b) paths in between (See Figure 6.2(a)). Such a path can be eliminated from consideration.

2. A *compound type (ii)(b) path* - a compound path with an even number of edges with both end vertices being project clones. This path will contain one or more type (ii)(b) paths joined together. Such a path can also be eliminated from consideration as its end edges are incident to exposed project clones.

3. A *compound type (iii)(a) path* - a compound path with an odd number of edges with both end edges being matched in $C(M_k)$. This path will contain a type (iii)(a) path at one end with zero or more type (ii)(b) paths joined to it (See Figure 6.2(b)). We will consider these paths for elimination later in this proof.

4. A *compound type (iii)(b) path* - a compound path with an odd number of edges with both end edges being matched in $C(M''_{k+1})$. This path will contain a type (iii)(b) path at one end with zero or more type (ii)(b) paths joined to it (See Figure 6.2(c)). We will consider these paths for elimination later in this proof.

Eliminating compound paths: At this stage we are left with only compound type (iii)(a) and compound type (iii)(b) paths in X'. These paths, if considered independently decrease and increase the size of $C(M_k)$ by 1 respectively. Since $|C(M''_{k+1})|= |C(M_k)|+1$ then there are q type (iii)(a) paths and $(q+1)$ type (iii)(b) paths. Consider some compound type (iii)(b) path D' and some compound type (iii)(a) path D''. Then we can consider the combined effect of augmenting $C(M_k)$ or $C(M''_{k+1})$ along $D' \cup D''$. Suppose that $\rho((D' \cup D'') \cap C(M''_{k+1})) \succ_L \rho((D' \cup D'') \cap C(M_k))$. A new matching $C(M''_k)$ in G' of cardinality k can be created by augmenting $C(M_k)$ along $D' \cup D''$. Since the upper quota constraints on the lecturers involved are not violated after creating $C(M''_k)$ from $C(M_k)$, then $C(M''_k)$ is also a valid SPA matching in I'. Thus $\rho(C(M''_k)) \succ_L \rho(C(M_k))$ which is a contradiction to the fact that $C(M_k)$ is a greedy k-matching in I'. A similar contradiction (to the fact that $C(M''_{k+1})$ is a greedy $(k+1)$-matching in I') exists if we assume $\rho((D' \cup D'') \cap C(M_k)) \succ_L \rho((D' \cup D'') \cap C(M''_{k+1}))$. Thus $\rho((D' \cup D'') \cap C(M''_{k+1})) = \rho((D' \cup D'') \cap C(M_k))$. It follows that, considering D' and D'' together, the size and profile of the matching is unaffected as augment from $C(M_k)$ to $C(M''_{k+1})$ or vice versa and so both D' and D'' can be eliminated from consideration.

Generating an augmenting path in $N(I)$: Once all these eliminations have been done, since $|C(M''_{k+1})|= |C(M_k)|+1$ it is easy to see that there remains only one path P' left in X' which is a compound type (iii)(b) path. The path P' can then be transformed to a component D in $G(I)$ (where $G(I)$ is basically the undirected counterpart of $N(I)$ without capacities) by replacing all the project clones p_j^r $(1 \leq r \leq c_j)$ in P' with the original project p_j and, for every joined pair of project clones $(p_j^r, p_{j'}^{r'})$, adding the lecturer $l(p_j^r) = l(p_{j'}^{r'})$ in between them. Thus a project may now appear more than once in D. A lecturer may also appear more than once in D.

Consider some project $p_j \in D$ that appears more than once. Then let $P'' \subset P'$ be the path consisting of edges between the first and last occurrence of the p_j clones in P' (P'' corresponds to a collection of cycles belonging to D in $G(I)$ involving p_j). Thus P'' is of even length and both end projects of P'' are clones of the same project. Augmenting $C(M_k)$ or $C(M''_{k+1})$ along P'' will not violate the lecturer upper quota constraints or affect the size or profile of the matching obtained (again using the same arguments presented above). Thus P'' can be eliminated from consideration. Although this potentially breaks P' into two separate paths in $G(I')$ it still remains connected in $G(I)$. Similarly consider some lecturer $l_k \in D$ that appears more than once. Then let $P''' \subset P'$ be the path consisting of edges between the first and last occurrence of the l_k clones in P' (P''' corresponds to a collection of type (ii)(b) paths with project clones offered by l_k). Thus augmenting $C(M_k)$ or $C(M''_{k+1})$ along P''' will not violate the

Algorithm 6.1 `Greedy-max-spa`

Require: SPA instance I;
Ensure: return matching M;
1: define flow network $N(I) = \langle G, c \rangle$;
2: define empty flow f;
3: **loop**
4: $P =$ `Get-max-aug`$(N(I), f)$;
5: **if** $P \neq null$ **then**
6: augment f along P;
7: **else**
8: return $M(f)$;

lecturer upper quota constraints or affect the size or profile of the matching obtained (again using the same arguments presented above). Thus P''' can be eliminated from consideration. Doing the above steps continually for all projects and lecturers that occur more than once in D eventually yields a valid path in $G(I)$ in which all nodes are visited only once.

Finally we describe how the path D in $G(I)$, obtained after removing duplicate projects and lecturers, can be transformed to an augmenting path P in $N(I)$ (i.e. we establish the direction of flow from v_s to v_t through P in $N(I)$). Firstly we add the edge (v_s, s_i) to P where s_i is the exposed student in D. Next for every edge $(s_{i'}, p_{j'}) \in M''_{k+1} \cap D$ we add a forward edge $(s_{i'}, p_{j'})$ to P. Also for every edge $(s_{i''}, p_{j''}) \in M_k \cap D$ we add a backward edge $(p_{j''}, s_{i''})$ to P. Finally we add the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ to P where p_j^r is the end project vertex in D. Thus P is an augmenting path with respect to $f = f(M_k)$ in $N(I)$ such that if f' is the flow obtained when f is augmented along P then $M(f')$ is a greedy $(k+1)$-matching in $N(I)$. □

Lemma 6.3.2. *Let f be a flow in N and let $M_k = M(f)$. Suppose that M_k is a greedy k-matching. Let P be a maximum profile augmenting path with respect to f. Let f' be the flow obtained by augmenting f along P. Now let $M_{k+1} = M(f')$. Then M_{k+1} is a greedy $(k+1)$-matching.*

Proof. Suppose for a contradiction that M_{k+1} is not a greedy $(k+1)$-matching. By Lemma 6.3.1, there exists an augmenting path P' with respect to f such that if f' is the result of augmenting f along P' then $M'_{k+1} = M(f')$ is a greedy $(k+1)$-matching. Hence $\rho(M'_{k+1}) \succ_L \rho(M_{k+1})$. Since $\rho(M'_{k+1}) = \rho(M) + \rho(P')$ and $\rho(M_{k+1}) = \rho(M) + \rho(P)$, it follows that $\rho(P') \succ_L \rho(P)$, a contradiction to the assumption that P is a maximum profile augmenting path. □

The `Get-max-aug` algorithm shown in Algorithm 6.2 accepts a flow network $N(I)$ and flow f as input and finds an augmenting path of maximum profile relative to

f or reports that none exists. The latter case implies that $M(f)$ is already a greedy maximum matching. The method consists of three phases: an initialisation phase (lines 1 -11), the main phase which is a loop containing two other loops (lines 12 - 31) and a final phase (lines 32 - 39) where the augmenting path is generated and returned.

For each project p_j the `Get-max-aug` method maintains a variable $\rho(p_j)$ describing the profile of a partial augmenting path P' from some exposed student to p_j. It also maintains, for every project $p_j \in \mathcal{P}$, a pointer $pred(p_j)$ to the student or lecturer preceding p_j in P'. For every lecturer $l_k \in \mathcal{L}$ a pointer $pred(l_k)$ is also used to refer to any project preceding l_k in P'. Thus the final augmenting path produced will pass through each lecturer or project at most once. The initialisation phase of the method involves setting all *pred* pointers to **null** and ρ profiles to B_R^-. Next, the method seeks to find, for each project p_j, a partial augmenting path $((v_s, s_i), (s_i, p_j))$ from the source, through an exposed student s_i to p_j should one exist. In the presence of multiple paths satisfying this criterion, the path with the best profile (w.r.t. $\succ_L$) is selected. The variables $pred(p_j)$ and $\rho(p_j)$ are updated accordingly. Thus at the end of this phase $\rho(p_j)$ indicates the maximum profile of an augmenting path of length 2 via some exposed student to p_j should one exist. If such a path does not exist then $\rho(p_j)$ and $pred(p_j)$ remain B_R^- and **null** respectively.

In the main phase, the algorithm then runs $|f|$ iterations, at each stage attempting to increase the quality (w.r.t. $\succ_L$) of the augmenting paths described by the ρ profiles. Each iteration runs two loops. Each loop identifies cases where the flow through one edge in the network can be reduced in order to allow the flow through another to be increased while improving the profile of the projects involved. In both loops, the decision on whether to switch the flow between candidate edges is made based on an edge relaxation operation similar to that used in the Bellman-Ford algorithm for solving the single source shortest path problem in which edge weights may be negative. In the first loop, we seek to evaluate the gain that may be derived from switching the flow through a student from one project to another. Given an edge (s_i, p_k) with a flow of 1 in f and edge (s_i, p_j) with no flow in f, we define σ to be the resulting profile of p_j if the partial augmenting path ending at p_k is to be extended (via s_i) to p_j. Thus σ will become the new value of $\rho(p_j)$ should this extension take place. If $\sigma \succ_L \rho(p_j)$ (i.e. if the proposed profile is better than the current one), we extend the augmenting path to p_j and update $\rho(p_j) = \sigma$ and $pred(p_j) = s_i$.

In the second loop, we seek to evaluate the gain that may be derived from switching flow to some lecturer from one project to another. Given a lecturer l_k, let $P'_k \subseteq P_k$ be the set of projects offered by l_k with positive outgoing flow and $P''_k \subseteq P_k$ be the set of projects offered by l_k that are undersubscribed in $M(f)$. Then we seek to determine if an improvement can be obtained by switching a unit of flow from some project $p_j \in P'_k$

to some other project $p_m \in P''_k$. This is achieved by comparing the $\rho(p_j)$ and $\rho(p_m)$ profiles and updating $\rho(p_j) = \rho(p_m)$, $pred(p_j) = l_k$ and $pred(l_k) = p_m$ if $\rho(p_m) \succ_L \rho(p_j)$ where $\rho(p_m)$ represents the profile of a partial augmenting path that does not already pass through l_k (i.e., $pred(p_m) \neq l_k$). This means that the partial augmenting path ending at p_m can be extended further (via l_k) to p_j while improving its profile. The intuition is that, after augmenting along such a path, p_m gains an extra student while p_j loses one.

During the final phase, we iterate through all exposed projects and find the one with the largest profile with respect to $\succ_L$ (say p_q). An augmenting path is then constructed through the network using the *pred* values of the projects and lecturers and the matched edges in $M(f)$ starting from p_q. The generated path is returned to the calling algorithm. If no exposed project exists, the method returns **null**. We next show that `Get-max-aug` method produces such a maximum profile augmenting path in N with respect to f should one exist.

Lemma 6.3.3. *Given a* SPA *instance I, let f be a flow in $N = N(I)$ where $k = |f|$ is not the size of a maximum matching in I and $M(f)$ is a greedy k-matching in I. Algorithm* `Get-max-aug` *finds a maximum profile augmenting path in N with respect to f.*

Proof. Consider some project p_j in $\mathcal{P}$. For any q $(0 \leq q \leq k)$ and for any r $(0 \leq r \leq k)$, we define $\Phi_{2q+1,2r}(p_j)$ to be the maximum profile of any partial augmenting path with respect to f in N that starts at an exposed student, ends at p_j, and involves at most $2q + 1$ student-project edges and at most $2r$ project-lecturer edges. We represent the length of such a path using the pair $(2q+1, 2r)$. Thus $\Phi_{2k+1,2k}(p_j)$ gives the maximum profile of any partial augmenting path starting at an exposed student and ending at p_j. If such a path does not exist then $\Phi_{2k+1,2k}(p_j) = B_R^-$. Firstly we seek to show that after q iterations of the main loop of `Get-max-aug` where $0 \leq q \leq k$, $\rho_q(p_j) \succeq_L \Phi_{2q+1,2q}(p_j)$ for every project $p_j \in \mathcal{P}$ where $\rho_q(p_j)$ is the profile computed at p_j after q iterations of the main loop.

We prove this inductively. For the base case, let $q = 0$. Then $\Phi_{1,0}(p_m)$ is the maximum profile of any partial augmenting path of length $(1, 0)$ from an exposed student to project p_m. Hence, from the initialisation phase of `Get-max-aug`, $\rho_0(p_m) = \Phi_{1,0}(p_m)$ and thus $\rho_0(p_m) \succeq_L \Phi_{1,0}(p_m)$. For the inductive step, assume $1 \leq q \leq k$ and that the claim is true after the $(q-1)^{th}$ iteration (i.e. $\rho_{q-1}(p_m) \succeq_L \Phi_{2q-1,2q-2}(p_m)$ for any $p_m \in \mathcal{P}$). We will show that the claim is true for the q^{th} iteration (i.e. $\rho_q(p_m) \succeq_L \Phi_{2q+1,2q}(p_m)$).

For each project $p_m \in \mathcal{P}$ let $S'_m = \{s_i \in \mathcal{S} : (s_i, p_m) \in E \wedge f(s_i, p_m) = 0\}$ and for each lecturer $l_k \in \mathcal{L}$ let $P'_k = \{p_m \in \mathcal{P} : l_k = l(p_m) \wedge f(p_m, l_k) < c_m\}$. For each iteration

Algorithm 6.2 `Get-max-aug` (method for `Greedy-max-spa`)

Require: flow network $N(I) = \langle G, c\rangle$ where $G = (V, E)$, flow f where $M(f)$ is a greedy $|f|$-matching;

/* initialisation */
for project $p_j \in \mathcal{P}$ **do**
  $\rho(p_j) = B_R^-$;
  $pred(p_j) =$ **null**;
  for each exposed student $s_i \in S$ such that $p_j \in A_i$ **do**
    $\sigma = O_R + rank(s_i, p_j)$;
    if $\sigma \succ_L \rho(p_j)$ **then**
      $\rho(p_j) = \sigma$;
      $pred(p_j) = s_i$;
for lecturer $l_k \in \mathcal{L}$ **do**
  $pred(l_k) =$ **null**;
/* main phase */
for 1...$|f|$ **do**
  /* first loop */
  for each $(s_i, p_j) \in E$ where $f(s_i, p_j) = 0$ and $f(s_i, p_k) = 1$ for some $p_k \in A_i$ **do**
    $\sigma = \rho(p_k) - rank(s_i, p_k) + rank(s_i, p_j)$;
    if $\sigma \succ_L \rho(p_j)$ **then**
      $\rho(p_j) = \sigma$; $pred(p_j) = s_i$;
  /* second loop */
  for each lecturer $l_k \in \mathcal{L}$ **do**
    $\sigma = B_R^-$;
    $p_z =$ **null**;
    for each project $p_m \in P_k$ such that $l(p_m) = l_k \wedge f(p_m, l_k) < c_m$ **do**
      if $\rho(p_m) \succ_L \sigma$ **then**
        $\sigma = \rho(p_m)$;
        $p_z = p_m$;
    if $p_z \neq$ **null then**
      for each project $p_j \in P_k$ such that $l(p_j) = l_k \wedge f(p_j, l_k) > 0 \wedge p_j \neq p_z$ **do**
        $\rho(p_j) = \sigma$;
        $pred(p_j) = l_k$;
        $pred(l_k) = p_z$;
/* final phase */
$\rho = \max_{\succ_L}(\{B_R^-\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
if $\rho \succ_L B_R^-$ **then**
  $p_q = \arg\max_{\succ_L}(\{B_R^-\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
  $Q =$ path obtained by following $pred$ values and matched edges in $M(f)$ from p_q to an exposed student;
  return $\langle v_s\rangle$ ++ $reverse(Q)$ ++ $\langle l(p_q), v_t\rangle$; /*++ denotes concatenation*/
else
  return **null**;

of the main loop, we perform a relaxation step involving some student-project pair (s_i, p_m) where $s_i \in S'_m$ and/or a relaxation step involving some project-lecturer pair (p_m, l_k) where $p_m \in P'_k$. Consider some project p_m. If there does not exist a partial augmenting path from an exposed student to p_m, of length $\leq (2q+1, 2q-2)$ and with

a better profile than $\Phi_{2q-1,2q-2}(p_m)$, then $\Phi_{2q+1,2q-2}(p_m) = \Phi_{2q-1,2q-2}(p_m)$. Otherwise there exists a partial augmenting path from an exposed student to p_m of length at least $(2q+1, 2q-2)$ with a better profile than $\Phi_{2q-1,2q-2}(p_m)$. Such a path must contain a partial augmenting path from an exposed student to some project $p_{m'}$ such that:

$$\Phi_{2q+1,2q-2}(p_m) = \Phi_{2q-1,2q-2}(p_{m'}) + rank(s_i, p_m) - rank(s_i, p_{m'}).$$

where $s_i \in S'_m$ and $f(s_i, p_{m'}) = 1$. Thus we note the following identity involving $\Phi_{2q+1,2q-2}(p_m)$:

$$\begin{aligned}\Phi_{2q+1,2q-2}(p_m) = \max\nolimits_{\succ_L}\{\{\Phi_{2q-1,2q-2}(p_m)\} \cup \\ \{\Phi_{2q-1,2q-2}(p_{m'}) + rank(s_i, p_m) - rank(s_i, p_{m'}) : s_i \in S'_m \wedge f(s_i, p_{m'}) = 1\}\}.\end{aligned} \tag{6.1}$$

Let $\rho'_q(p_m)$ be the profile computed at p_m after the first sub-loop during the q^{th} iteration of the main loop of the `Get-max-aug` algorithm (i.e. at Line 19 during the q^{th} iteration). Then

$$\begin{aligned}\rho'_q(p_m) = \max\nolimits_{\succ_L}\{\{\rho_{q-1}(p_m)\} \cup \\ \{\rho_{q-1}(p_{m'}) + rank(s_i, p_m) - rank(s_i, p_{m'}) : s_i \in S'_m \wedge f(s_i, p_{m'}) = 1\}\}.\end{aligned} \tag{6.2}$$

By the induction hypothesis, $\rho_{q-1}(p_m) \succeq_L \Phi_{2q-1,2q-2}(p_m)$. Thus:

$$\begin{aligned}&\rho'_q(p_m) = \max\nolimits_{\succ_L}\{\{\rho_{q-1}(p_m)\} \cup \{\rho_{q-1}(p_{m'}) + rank(s_i, p_m) - rank(s_i, p_{m'}) : \\ &\qquad s_i \in S'_m \wedge f(s_i, p_{m'}) = 1\}\} \text{ (by equation 6.2).} \\ &\succeq_L \max\nolimits_{\succ_L}\{\{\Phi_{2q-1,2q-2}(p_m)\} \cup \{\Phi_{2q-1,2q-2}(p_{m'}) + rank(s_i, p_m) - rank(s_i, p_{m'}) : \\ &\qquad s_i \in S'_m \wedge f(s_i, p_{m'}) = 1\}\} \text{ (by the inductive hypothesis)} \\ &= \Phi_{2q+1,2q-2}(p_m). \text{ (by equation 6.1)}\end{aligned}$$

Therefore:

$$\rho'_q(p_m) \succeq_L \Phi_{2q+1,2q-2}(p_m). \tag{6.3}$$

Again, if there does not exist a partial augmenting path from an exposed student to p_m, of length $\leq (2q+1, 2q)$ and with a better profile than $\Phi_{2q+1,2q-2}(p_m)$, then $\Phi_{2q+1,2q}(p_m) = \Phi_{2q+1,2q-2}(p_m)$. Otherwise there exists a partial augmenting path from an exposed student to p_m of length $(2q+1, 2q)$ with a better profile than $\Phi_{2q+1,2q-2}(p_m)$. We can therefore note the following identity involving $\Phi_{2q+1,2q}(p_m)$:

$$\begin{aligned}\Phi_{2q+1,2q}(p_m) = \max\nolimits_{\succ_L}\{\{\Phi_{2q+1,2q-2}(p_m)\} \cup \\ \{\Phi_{2q+1,2q-2}(p_{m'}) : l_k = l(p_m) \wedge p_{m'} \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_t) = d_k^+\}\}.\end{aligned} \tag{6.4}$$

After the q^{th} iteration of the main loop has completed, we have:

$$\begin{aligned}\rho_q(p_m) = \max_{\succ_L}\{\{\rho'_q(p_m)\} \cup \\ \{\rho'_q(p_{m'}) : l_k = l(p_m) \wedge p_{m'} \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_t) = d_k^+\}\}.\end{aligned} \tag{6.5}$$

We observe that the extra condition ($pred(p_m) \neq l_k$) in Line 23 of the second loop, does not affect the correctness of equation 6.5. Suppose $pred(p_m) = l_k$, then $\rho(p_m)$ must have been updated during the q^{th} iteration of the second loop (or during a previous iteration and has remained unchanged) by some project profile $\rho(p'_j)$. Thus setting $\rho(p_j) = \rho(p_m)$ and $pred(l_k) = p_m$ would be incorrect as p'_j is now the source of $\rho(p_m)$ and not p_m. Moreover if indeed $\rho(p_m) = \rho(p'_j) \succ_L \rho(p_j)$ then p'_j would be encountered later on during the iteration of the second loop.

$$\begin{aligned}&\rho_q(p_m) = \max_{\succ_L}\{\{\rho'_q(p_m)\} \cup \\ &\qquad \{\rho'_q(p_{m'}) : l_k = l(p_m) \wedge p_{m'} \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_t) = d_k^+\}\}. \\ &\succeq_L \max_{\succ_L}\{\{\Phi_{2q+1,2q-2}(p_m)\} \cup \{\Phi_{2q+1,2q-2}(p_{m'}) : l_k = l(p_m) \wedge \\ &\qquad p_{m'} \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_t) = d_k^+\}\} \text{ (from equation 6.3)} \\ &= \Phi_{2q+1,2q}(p_m) \text{ (by equation 6.4).}\end{aligned}$$

Therefore:

$$\rho_q(p_m) \succeq_L \Phi_{2q+1,2q}(p_m).$$

But any partial augmenting path from an exposed student to p_j with respect to flow f can have length at most $(2k+1, 2k)$. Thus $\rho(p_j) = \Phi_{2k+1,2k}(p_j)$ after k iterations of the main loop.

Finally we show that a partial augmenting path P' (and subsequently a full augmenting path) can be constructed by following the *pred* values of projects and lecturers and the matched edges in $M(f)$ starting from some exposed project p_j with the maximum $\rho(p_j)$ profile, and ending at some exposed student (i.e. we show that such a path is continuous and contains no cycle).

Suppose for a contradiction that such a path P' contained a cycle C. Then at some step X during the execution of the algorithm, C would have been formed when, for some project p_j, either (i) $pred(p_j)$ was set to some student s_i or (ii) $pred(p_j)$ was set to some lecturer l_k. Let P'' be any path in $N(I)$. We may extend our definitions for the profile of a matching and a partial augmenting path to cover the profile of any path in $N(I)$ as follows:

$$\begin{aligned}\rho(P'') = O_R + \sum\{rank(s_i, p_j) : (s_i, p_j) \in P'' \cap E_2 \wedge f(s_i, p_j) = 0\} - \\ \sum\{rank(s_i, p_j) : (p_j, s_i) \in P'' \cap E_2 \wedge f(s_i, p_j) = 1\}.\end{aligned}$$

Considering case (i) let $p_m = M(s_i)$. Also let $\rho'(p_j)$ and $\rho(p_j)$ be the profiles of partial augmenting paths from some exposed student to p_j before and after step X respectively. Then $\rho(p_j) \succ_L \rho'(p_j)$. Also $\rho(p_j) = \rho(p_m) + rank(s_i, p_j) - rank(s_i, p_m)$, i.e., $\rho(p_j) = \rho(p_m) + \rho(P'')$ where $P'' = \{(s_i, p_j), (s_i, p_m)\}$. Since we can also trace a path through all the other projects in C (using *pred* values and matched edges) from p_m to p_j, it follows that $\rho(p_m) = \rho'(p_j) + \rho(C \backslash \{(s_i, p_j), (s_i, p_m)\})$. Thus $\rho(p_j) = \rho'(p_j) + \rho(C)$. Note that $\rho(C) = \rho(C' \backslash M) - \rho(C' \cap M)$ and $C' = C \cap E_2$ is the set of edges in C involving only students and projects. As $\rho(p_j) \succ_L \rho'(p_j)$, it follows that $\rho(C' \backslash M) \succ_L \rho(C' \cap M)$. But since $|C' \backslash M| = |C' \cap M|$, and lecturer capacities are clearly not violated by the algorithm, a new matching $M' = M \oplus C'$ can be generated such that $\rho(M') \succ_L \rho(M)$ and $|M'| = |M| = |f|$, a contradiction to the fact that M is a greedy $|f|$-matching in I.

Considering case (ii) let $p_m = pred(l_k)$. As before let $\rho'(p_j)$ and $\rho(p_j)$ be the profiles of partial augmenting paths from some exposed student to p_j before and after step X respectively. Then $\rho(p_j) \succ_L \rho'(p_j)$. Also $\rho(p_j) = \rho(p_m)$. Since we can also trace a path through all the other projects in C (using *pred* values and matched edges) from p_m to p_j, it follows that $\rho(p_m) = \rho'(p_j) + \rho(C \backslash \{(p_j, l_k), (p_m, l_k)\}) = \rho'(p_j) + \rho(C)$. Thus $\rho(p_j) = \rho'(p_j) + \rho(C)$. Note that $\rho(C) = \rho(C' \backslash M) - \rho(C' \cap M)$ and $C' = C \cap E_2$ is the set of edges in C involving only students and projects. As $\rho(p_j) \succ_L \rho'(p_j)$, it follows that $\rho(C' \backslash M) \succ_L \rho(C' \cap M)$. A similar argument to the one presented above shows a contradiction to the fact that M is a greedy $|f|$-matching in I. □

From Lemmas 6.3.1, 6.3.2 and 6.3.3, we can conclude that the algorithm `Greedy-max-spa` finds a greedy maximum matching given a SPA instance. Concerning the complexity of the algorithm, the main loop calls `Get-max-aug` η times where η is the size of a maximum cardinality matching in I. The first phase of `Get-max-aug` performs $O(m_2)$ profile comparison operations and $O(n_3)$ initialisation steps for the lecturer *pred* values where $m_2 = |E_2|$, $n_3 = |\mathcal{L}|$, and each profile comparison step requires $O(R)$ time. The loop in the main phase of `Get-max-aug` runs k times where k is the value of the flow obtained at that time. The first and second loops perform $O(m_2)$ and $O(n_2)$ relaxation steps respectively where $n_2 = |\mathcal{P}|$ and each relaxation step requires $O(R)$ time to compare profiles. The final phase of the algorithm performs $O(n_2)$ profile comparisons, each also taking $O(R)$ time. Thus the overall time complexity of the `Get-max-aug` method is $O(m_2R + n_3 + kR(m_2 + n_2) + n_2R) = O(kR(m_2))$. Thus the overall time complexity of the `Greedy-max-spa` algorithm is $O(n_1^2 R m_2)$.

When considering the additional factor of $O(R)$ due to arithmetic on edge weights of $O(n_1^R)$ size, Orlin's algorithm for finding a minimum cost maximum flow in $N(I)$ runs in $O(Rm_2^2 \log(n_1 + n_2) + Rm_2(n_1 + n_2)\log^2(n_1 + n_2))$ time [96]. Suppose $n_1 \geq n_2$. Then Orlin's algorithm runs in $O(Rm_2^2 \log n_1 + n_1 R m_2 \log^2 n_1)$ time. If the first term

of Orlin's runtime is larger than the second then our algorithm is slower by a factor of $\frac{n_1^2}{m_2 \log n_1} \leq \frac{n_1}{\log n_1}$ as $m_2 \geq n_1$. If the second term of Orlin's runtime is larger than the first then our algorithm is slower by a factor of $\frac{n_1}{\log^2 n_1} \leq \frac{n_1}{\log n_1}$.

Now suppose $n_2 > n_1$. Then Orlin's algorithm runs in $O(Rm_2^2 \log n_2 + n_2 Rm_2 \log^2 n_2)$ time. If the first term of Orlin's runtime is larger than the second then our algorithm is slower by a factor of $\frac{n_1^2}{m_2 \log n_2} \leq \frac{n_1}{\log n_2} \leq \frac{n_1}{\log n_1}$ as $m_2 \geq n_1$ and $n_2 > n_1$. If the second term of Orlin's runtime is larger than the first then our algorithm is slower by a factor of $\frac{n_1^2}{n_2 \log^2 n_2} \leq \frac{n_1}{\log^2 n_2} \leq \frac{n_1}{\log n_1}$ as $n_2 > n_1$.

So our algorithm is slower than Orlin's by a factor of $\frac{n_1}{\log n_1}$ in all cases. A straightforward refinement of our algorithm can be made by observing that if no profile is updated during an iteration of the main loop, then no further profile improvements can be made and we can terminate the main loop at this point. We conclude with the following theorem.

Theorem 6.3.4. *Given a* SPA *instance I, a greedy maximum matching in I can be obtained in $O(n_1^2 Rm_2)$ time.*

6.4 Generous maximum matchings in SPA

Analogous to the case for greedy maximum matchings, generous maximum matchings can also be found by modelling SPA as a network flow problem. Given a SPA instance I we define the following terms relating to partial augmenting paths in $N(I)$. For each project $p_j \in \mathcal{P}$, we define the *minimum profile of a partial augmenting path* from v_s through an exposed student to p_j with respect to $\prec_R$, denoted $\Phi'(p_j)$, as follows:

$$\Phi'(p_j) = \min_{\prec_R}\{\rho(P') : P' \text{ is a partial augmenting path from } v_s \text{ to } p_j\}.$$

If a partial augmenting path P' ending at project p_j can be extended to an augmenting path P by adding edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ then such an augmenting path is called a *minimum profile augmenting path* if $\rho(P) = \min_{\prec_R}\{\Phi'(p_j) : p_j \in \mathcal{P}\}$. A similar approach to that used to find a greedy maximum matching can be adopted in order to find a generous maximum matching. The main `Greedy-max-spa` algorithm will remain unchanged (we will call it `Generous-max-spa` for convenience) as the intuition remains to successively find larger generous k-matchings until a generous maximum matching is obtained. We however make slight changes to the `Get-max-aug` algorithm in order to find a minimum profile augmenting path in the network should one exist (the resulting algorithm is then known as `Get-min-aug`). The changes are as follows. (i) We replace all occurrences of left domination $\succ_L$ with right domination $\prec_R$. (ii)

We also replace all occurrences of negative infinity profile B_R^- with a positive infinity profile B_R^+. (iii) Finally we replace both max functions (in lines 33 and 35) with the min function. Analogous statements and proofs of Lemmas 6.3.1, 6.3.2 and 6.3.3 exist in this context. Thus we may conclude with the following theorem concerning the `Generous-max-spa` algorithm.

Theorem 6.4.1. *Given a* SPA *instance I, a generous maximum matching in I can be obtained in $O(n_1^2 R m_2)$ time.*

6.5 Integer Programming models for SPA

6.5.1 Introduction

As mentioned earlier, IP is a common technique for solving matching problems like SPA. In this section we construct IP models for SPA. We denote S_j as the set of students who applied for project p_j. We partition the projects on each student's preference list into sets based on their rank. For student s_i, we define $A_i^r = \{p_j : rank(s_i, p_j) = r\}$. In the following subsections we present two IP models for MAX SPA which vary from each other depending on the structure of their objective functions. For both models we define the variable $x_{i,j} \in \{0, 1\}$ which corresponds to project p_j on student s_i's preference list. Given a solution to any of these IP models, we construct a matching M as follows. If $x_{i,j} = 1$ we make $(s_i, p_j) \in M$ otherwise we make $(s_i, p_j) \notin M$.

6.5.2 Model with exponential coefficients (`Model1`)

Figure 6.3 shows our first model (`Model1`) for finding a greedy maximum matching given a SPA instance. Constraint 1 ensures that each student is matched to at most one project while Constraint 2 ensures that no project exceeds its upper quota. Constraint 3 ensures that no lecturer exceeds her upper quota. The objective function is taken from the transformation of the *House Allocation problem with Ties* (HAT) to the *Assignment Problem* (AP) presented in [85] for finding a greedy maximum matching given a HAT instance. It employs a set of steeply decreasing variable coefficients which are exponentially large in the number of students. The first term in each coefficient ensures that a maximum matching is always obtained. The second term ensures that it is always beneficial to have a single student matched to their kth choice project than have n_1 students matched to their $(k+1)$th choice project. A similar objective function can be constructed for finding a generous maximum matching by having coefficient $(n_1^R - n_1^{r-1})$ for each variable $x_{i,j}$ where $p_j \in A_i^r$.

$$\max \sum_{s_i \in \mathcal{S}} \sum_{r=1}^{R} \sum_{p_j \in A_i^r} (n_1^R + n_1^{R-r})\ x_{i,j}$$

subject to:

1. $\sum_{p_j \in A_i} x_{i,j} \leq 1 \qquad \forall\ s_i \in \mathcal{S}$

2. $\sum_{s_i \in S_j} x_{i,j} \leq c_j^+ \qquad \forall\ p_j \in \mathcal{P}$

3. $\sum_{p_j \in P_k} \sum_{s_i \in S_j} x_{i,j} \leq d_k^+ \qquad \forall\ l_k \in \mathcal{L}$

$x_{i,j} \in \{0, 1\}$

Figure 6.3: `Model1`: IP model for finding a greedy maximum matching given a SPA instance

$$\max \sum_{s_i \in \mathcal{S}} \sum_{p_j \in A_i} x_{i,j}$$

subject to:

1. $\sum_{p_j \in A_i} x_{i,j} \leq 1 \qquad \forall\ s_i \in \mathcal{S}$

2. $\sum_{s_i \in S_j} x_{i,j} \leq c_j^+ \qquad \forall\ p_j \in \mathcal{P}$

3. $\sum_{p_j \in P_k} \sum_{s_i \in S_j} x_{i,j} \leq d_k^+ \qquad \forall\ l_k \in \mathcal{L}$

$x_{i,j} \in \{0, 1\}$

Figure 6.4: `Model2`: IP model for finding a maximum matching given a SPA instance

6.5.3 Model with hierarchical objectives (`Model2`)

An alternative IP model can be constructed by considering a set of hierarchical objective functions. As shown in Figure 6.4, we start with an initial IP model (`Model2`) with the same constraints as those presented in Figure 6.3. The initial objective function would seek to maximise the number of students who are assigned a project. Once `Model2` is solved, the optimal value (which we call η) can then be used as a bound on the size of matchings found in subsequent models. This is achieved by adding the following constraint to all subsequent models (we call this a *size constraint*).

$$4. \qquad \sum_{s_i \in \mathcal{S}} \sum_{p_j \in A_i} x_{i,j} \geq \eta$$

Algorithm 6.3 `Hierarchy-spa-ip`

Require: SPA instance I;
Ensure: return matching M;
1: construct `Model2` instance J from I;
2: solve J;
3: **for** $1 \leq r \leq R$ **do**
4: set rank-r objective function;
5: add rank-$(r-1)$ constraint to J;
6: solve J;
7: return matching M derived from optimal solution to J;

This ensures that subsequent matchings found would always be maximum. In order to achieve the greedy optimisation criteria, a series of R IP models are then created (and solved) by adding extra constraints and updating the objective function, where R is the maximum length of any student's preference list. The aim is to optimise the number of students matched to their first-choice projects, and subject to that, the number of students matched to their second-choice projects, etc., in accordance with the greedy optimisation criterion. The first model is created by adding the size constraint and changing the objective function to maximise the number of students assigned to their first-choice project. For subsequent models, the objective function is changed in order to maximise the number of students matched to their rth choice project where $1 < r \leq R$. At stage r of the loop, the objective function would be as follows (we call this a *rank-r objective function*)

$$\max \sum_{s_i \in \mathcal{S}} \sum_{p_j \in A_i^r} x_{i,j}$$

In addition to the size constraint, we also add constraints to ensure that the results obtained by previous iterations are preserved. Thus at stage r of the loop, in addition to the basic constraints defined in Figure 6.4, the following constraints would also be enforced. Where Constraint $5.r'$ enforces a bound $\lambda^{r'}$ on the number of students assigned to their r'th choice project. We call this a *rank-r' constraint*.

$$4. \qquad \sum_{s_i \in \mathcal{S}} \sum_{p_j \in A_i} x_{i,j} \geq \eta$$

$$5.r'. \qquad \sum_{s_i \in \mathcal{S}} \sum_{p_j \in A_i^{r'}} x_{i,j} \geq \lambda^{r'} \qquad \forall\ 1 \leq r' \leq r-1$$

Algorithm 6.3 describes the process. For simplicity of presentation we define the size constraint as a rank-0 constraint. Although this approach would mean multiple calls to

the IP solver and may suffer from scalability issues, it does not employ the exponentially large variable coefficients seen in `Model1`.

6.6 Empirical evaluation

6.6.1 Introduction

The `Greedy-Max-Spa` and `Generous-Max-Spa` algorithms were implemented in Java and evaluated empirically. The SPA IP models were encoded and solved using the IBM CPLEX 12.5.1 IP solver. In this section, we present results from empirical evaluations carried out on the IP encodings and algorithm implementations using both real-world and randomly-generated data. Results from the implemented algorithms were compared with those produced by an IP model of SPA in order to improve our confidence in the correctness of both implementations. We also investigate the feasibility issues that will be faced if a Min-Cost-Max-Flow (MCMF) approach (as suggested in [116]) is to be used when solving instances of SPA involving large numbers of students and projects, or where students have long preference lists. Other experiments carried out involve varying certain properties of the randomly-generated SPA instances while measuring the runtime of the algorithms and the size, degree and cost of the matchings produced.

An instance generator was used to construct random SPA instances which served as input for both the algorithm implementations and the IP encoding. This generator can be configured to vary certain properties of the SPA instances produced as follows:

1. The number of students n_1 (with a default value of $n_1 = 100$). The number of projects and lecturers are set to $n_2 = 0.3n_1$ and $n_3 = 0.2n_1$ respectively.

2. The minimum R_{min} and maximum R_{max} length of any student's preference list (with default values $R_{min} = R_{max} = 10$).

3. The popularity λ of the projects, as measured by the ratio between the number of students applying for one of the most popular projects and the number of students applying for one of the least popular projects (default value of 10 which is in line with the popularity ratios observed in real-world SPA datasets).

4. The total capacity of the projects $C_{\mathcal{P}}$ and lecturers $C_{\mathcal{L}}$. These capacities were not divided evenly amongst the projects and lecturers involved (default values are $C_{\mathcal{P}} = 1.2n_1$ and $C_{\mathcal{L}} = 1.2n_1$).

5. The tie density t_d ($0 \leq t_d \leq 1$) of the students' preference list. This is the probability that some project is tied with the one preceding it on some student's preference list (default value is $t_d = 0$).

6. The total project and lecturer lower quotas $L_{\mathcal{P}}$ and $L_{\mathcal{L}}$ respectively. These lower quotas were divided evenly amongst the projects and lecturers involved (default values are $L_{\mathcal{P}} = L_{\mathcal{L}} = 0$).

We also created SPA instances from anonymised data obtained from previous runs of the student-project allocation scheme at the School of Computing Science, University of Glasgow and solved them using the implemented algorithm and IP encoding. The default values for the randomly generated SPA instances somewhat vary from the real world datasets. This is because our experiments aim to investigate certain properties of the problems and algorithms which may not be highly visible in the real-world datasets. We measured the runtime taken by the algorithms and IP encodings as well as the size, cost and degree of the matchings obtained. Experiments were carried out on a Windows machine with 4 Intel(R) Core(R) i5-2400 CPUs at 3.1GHz and 8GB RAM.

In the following subsections we present results obtained from the empirical evaluations carried out. In Section 6.6.2 we present the results of correctness tests carried out by comparing results obtained from the IP models of SPA and from the implemented algorithms. In Section 6.6.3 we demonstrate when the MCMF approach becomes infeasible in practice. In Section 6.6.4 we present results from running the algorithms against real-world SPA instances. In Section 6.6.5 we vary certain properties of randomly generated SPA instances while measuring the runtime of the algorithms and the size, degree and cost of the matchings produced. We make some concluding remarks in Section 6.6.6

6.6.2 Testing for correctness

Although the `Greedy-Max-Spa` and `Generous-Max-Spa` algorithms have been proven to be correct (See Theorems 6.3.4 and 6.4.1), bugs may still exist in the implementations. In order to improve our confidence in any empirical results obtained as part of an experimental evaluation of the algorithms' performance, we compared results from the implemented algorithms with those obtained from IP models of SPA described in Section 6.5. For each value of n_1 in the range $n_1 \in \{20, 40, 60, ..., 200, 300, 400, ..., 1000\}$, 10,000 random SPA instances were generated and solved using both methods. For each SPA instance generated, $R_{min} = R_{max} = 10$ (henceforth we refer to $R_{min} = R_{max}$ as R). The profiles of the resulting matchings were then compared and observed to be identical for

all the instances generated. The resulting matchings were also tested to ensure they obeyed all the upper quota constraints for lecturers and projects. For `Model1`, due to the exponentially large variable coefficients, instance sizes were limited to $n_1 < 60$ due to floating-point precision errors when comparing values larger than 60^{10}. Thus, for the correctness tests, the second IP model described in Algorithm 6.3 was employed. This allowed for reasonably larger SPA instances to be solved.

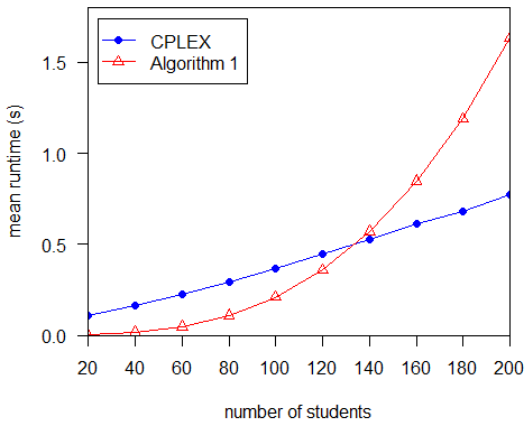

Figure 6.5: Mean runtime vs n_1

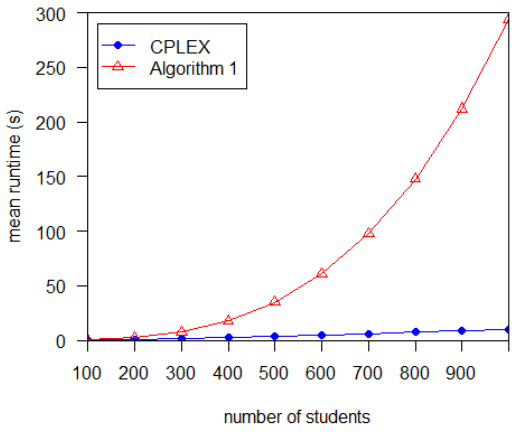

Figure 6.6: Mean runtime vs n_1

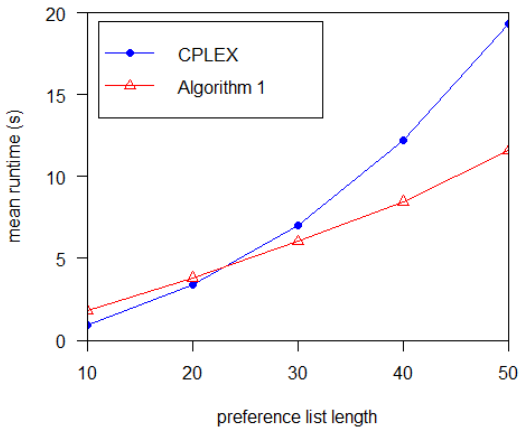

Figure 6.7: Mean runtime vs R

The mean runtimes (taken over 10,000 randomly generated instances) for finding greedy maximum matchings using both the IP model (`Model2`) and the `Greedy-max-spa` implementation were also measured. The results are shown in Figures 6.5, 6.6 and 6.7. Figure 6.5 shows the mean time taken to find an optimal solution as we increase the number of students from $n_1 = 20$ to $n_1 = 200$ in steps of 20. For the range of n_1 $20 \leq n_1 \leq 130$ the results show that the Java implementation of the `Greedy-max-spa` algorithm performs better than the IP encoding when solved using CPLEX. However for n_1 in the range $n_1 \geq 140$ we observe that the CPLEX runtime is smaller than the `Greedy-max-spa` algorithm. The results show that the CPLEX curve rises a lot

less steeply than that of the Java implementation. This trend is further highlighted in Figure 6.6 when even larger instance sizes are solved using both methods (instance sizes increasing from $n_1 = 100$ to $n_1 = 1000$ in steps of 100).

These results show how well the IP encodings for SPA problems (and perhaps the CPLEX solver in particular) can perform in practice for SPA instances with relatively short preference lists ($R = 10$) (despite the theoretically exponential time complexity of the IP branch and cut algorithms). As for the implementation of the `Greedy-max-spa` algorithm, the results are in line with our expectations considering the complexity of the algorithm and the technologies used in the implementation. Figure 6.7 shows what happens if we consider problem instances with longer preference lists. For each value of R in the range $R \in \{10, 20, 30, 40, 50\}$, we solve $1,000$ randomly generated SPA instances where $n_1 = 200$. We observe the runtime of the IP model rises more steeply than the runtime for `Greedy-max-spa`. For SPA instances with preference list lengths $R > 24$, `Greedy-max-spa` runs faster than the IP implementation and the slopes of both curves suggests the difference will increase as R increases. This is one scenario that highlights a shortcoming of the IP approach thus justifying the development of the efficient algorithms for these problems.

These correctness tests show that our implementations are likely to be correct and that large instances of the profile-based SPA problems can solved efficiently in practice.

6.6.3 Feasibility analysis of the MCMF approach

We implemented an algorithm for finding a minimum cost maximum flow in a given network. As stated in Section 2.6, by the appropriate assignment of edge costs/weights in the underlying network $N(I)$ of a SPA instance I, a minimum cost maximum flow algorithm can be used to find greedy and generous maximum matchings in I. We argued that this approach (as described in [5, 116]) would be infeasible due to the floating-point inaccuracies caused by the assignment of exponentially large edge costs/weights in the network. In this section we investigate this claim experimentally and demonstrate the feasibility issues that arise when using various Java data types to represent these edge weights.

The cost functions presented in [5, 116] and in the first IP model (`Model1`) presented in Figure 6.3 require a maximum cost maximum cardinality matching to be found. We therefore derive modified cost functions that require a minimum cost maximum flow algorithm to be used in order to find greedy and generous maximum matchings. For finding greedy maximum matchings we set the cost of an edge between a student s_i and a project p_j as $n_1^{R-1} - n_1^{R-k}$ where $k = rank(s_i, p_j)$. For finding generous maximum

matchings we set the cost of an edge between a student s_i and a project p_j as n_1^{k-1} where $k = rank(s_i, p_j)$. The cost for all other edges in the network are set to 0. For the MCMF approach, we define an instance as infeasible if the matching produced is not optimal with respect to the greedy or generous criteria (when compared with optimal results produced by the `Greedy-Max-Spa` and `Generous-Max-Spa` algorithms and CPLEX).

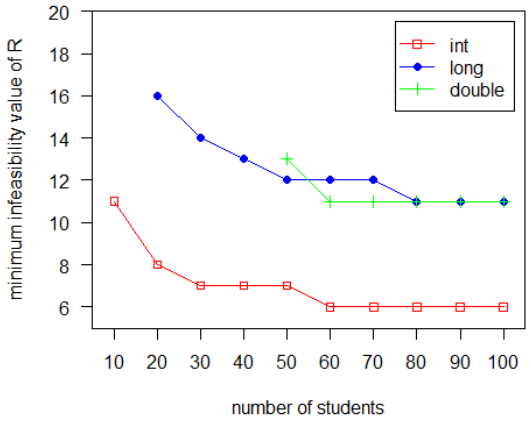

Figure 6.8: MCMF feasibility results

Figure 6.8 shows the feasibility results using three Java data types. For each value of n_1 (number of students) in the range $n_1 \in \{10, 20, 30, ..., 100\}$ and for each value of R (length of each student's preference list) in the range $R \in \{5, 6, ..., \min\{20, 1.2n_1\}\}$, we generated 1000 random SPA instances and solved them using the MCMF approach and the `Greedy-Max-Spa` algorithm. The graph shows the value of R at which infeasible solutions were first encountered. As expected, this number drops as we increase the instance size. Due to their greater precision that the `long` and `double` data types (when compared with `int`), we see that they handle much larger instances before encountering infeasibility issues. All instances tested for $n_1 = 10$ when using `long` and $n_1 \in \{10, 20, 30, 40\}$ when using `double` produced optimal matchings. This is probably because we do not yet encounter range errors (in the case of `long`) and precision errors (in the case of `double`) when solving these instances. The relatively low values of R and n_1 observed where infeasibility prevails (e.g., $n_1 = 60, R = 6$ for the `int` type) reinforces our argument that approaches based on MCMF which employ these exponentially large edge weights are not scalable.

6.6.4 Real-world data

SPA instances derived from anonymised data obtained from previous runs of the student-project allocation scheme at the School of Computing Science, University of Glasgow were created and solved using the `Greedy-Max-Spa` and `Generous-Max-Spa` implemen-

tations as well as the IP encodings. This section discusses some of the results obtained. Table 6.1 shows the properties of the generated SPA instances (with lecturer capacities not being considered in the 07/08 and 08/09 sessions) and Table 6.2 shows details of various profile-based optimal matchings found.

Session	n_1	n_2	n_3	R	$C_{\mathcal{P}}$	$C_{\mathcal{L}}$
14/15	51	147	37	6	147	80
13/14	51	155	40	5	155	77
12/13	38	133	34	5	133	63
11/12	31	103	26	5	103	62
10/11	34	63	29	5	63	66
09/10	32	102	28	5	102	72
08/09*	37	56	-	5	56	56
07/08*	35	61	-	5	61	61

Table 6.1: Real-world SPA instances

Session	$\|M\|$	Greedy		Generous		Min-Cost	
		Profile	Cost	Profile	Cost	Profile	Cost
14/15	51	$(30, 7, 1, 5, 5, 3)$	110	$(16, 16, 9, 6, 4)$	119	$(28, 11, 3, 5, 2, 2)$	101
13/14	51	$(26, 7, 4, 6, 8)$	116	$(15, 18, 9, 6, 3)$	117	$(23, 12, 5, 6, 5)$	111
12/13	38	$(26, 6, 3, 2, 1)$	60	$(21, 13, 4)$	59	$(23, 11, 3, 1)$	58
11/12	31	$(22, 6, 2, 1)$	44	$(20, 9, 2)$	44	$(20, 9, 2)$	44
10/11	34	$(25, 4, 3, 1, 1)$	51	$(21, 9, 4)$	51	$(24, 5, 4, 1)$	50
09/10	32	$(23, 4, 2, 2, 1)$	50	$(19, 10, 3)$	48	$(20, 9, 2, 1)$	48
08/09*	37	$(26, 6, 2, 1, 2)$	58	$(23, 11, 3)$	54	$(23, 11, 3)$	54
07/08*	35	$(20, 9, 5, 0, 1)$	58	$(17, 14, 4)$	57	$(17, 14, 4)$	57

Table 6.2: Real-world SPA results

The results demonstrate a drawback in adopting the greedy optimisation criterion, namely that some students may have projects that are far down their preference lists. In all but the 2011/2012 session, at least one student had her worst choice project in a greedy maximum matching. In the 2013/2014 session the number of students with their worst choice project is reasonably high and so the greedy maximum matching would probably not be selected for that year.

The degree of generous maximum matchings are usually less than the others (obviously they are never greater). This is usually an attractive property in such matching schemes. In all the years considered apart from the 2013/2014 session all students got their third choice project or better in the generous maximum matchings produced. However in the 2013/2014 session applying the generous optimality criteria did not improve on the degree of the matchings produced.

One of the major advantages of the minimum cost maximum matching optimality criteria is that in a certain sense it is more "egalitarian". Minimising the overall cost of the matchings produced is also a very natural objective. It may be considered a disadvantage if matchings obtained by adopting the profile-based optimality criteria have significantly larger costs than the minimum obtainable cost. However, from the results obtained on these real-world datasets, there is very little difference between the costs of the greedy and generous maximum matchings and the minimum obtainable costs (except, once again, for the 2013/2014 session). Thus we can choose one of the profile-based optimal matchings with some confidence that it is "almost" of minimum cost. In Section 6.6.5 we consider these differences on multiple randomly generated SPA instances.

6.6.5 Random data

6.6.5.1 Introduction

This section discusses some of the results obtained by varying certain properties of the randomly generated SPA instances and measuring the cost, size and degree of the matchings produced. For each instance generated we found a greedy maximum matching, a generous maximum matching and a minimum cost maximum matching.

6.6.5.2 Varying the number of students

Keeping R constant, we investigated the effects of increasing the number of students n_1 (and by implication n_2, n_3, $C_{\mathcal{P}}$ and $C_{\mathcal{L}}$ using the default dependencies listed in Section 6.6.1) on the degree, cost and size of the matchings produced as well as the time taken to find these matchings. For each value of n_1 in the range $n_1 \in \{100, 200, 300, ..., 700\}$ we generated and solved 100 random SPA instances.

Figure 6.9 shows the way the mean degree varies as we increase the number of students. The mean degrees of the greedy maximum matchings are the highest of the three with mean values ≥ 8 for $n_1 \geq 200$. As expected generous maximum matchings have the smallest degree, which rises slowly from about 4.8 to 6.5. An interesting observation is that the mean degree does not steeply rise as we increase the number of students. Also the mean degree for the minimum cost maximum matching is closer to the generous maximum matching degree than that of the greedy maximum matching. This is probably due to the fact that the cost function (*rank* in this case) is greater for higher degrees than lower ones, so, in some way, by minimising the cost, we are also seeking matchings with fewer students matched to projects that are father down their preference lists (i.e. have higher ranks).

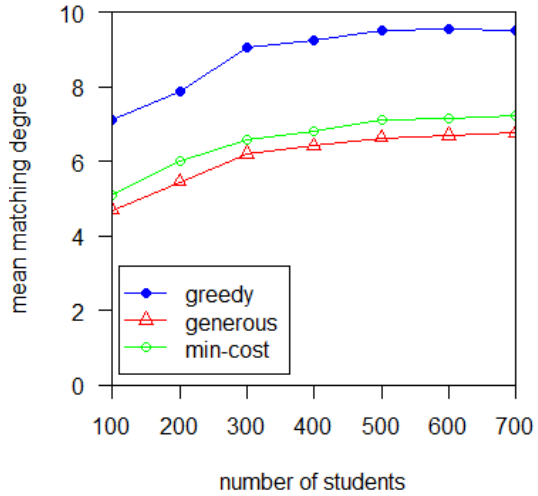

Figure 6.9: Mean matching degree vs n_1

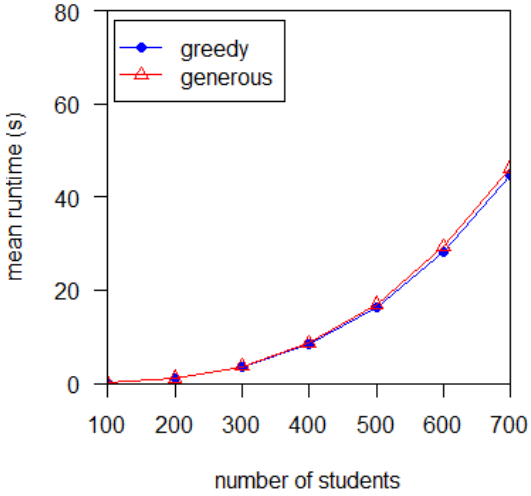

Figure 6.10: Mean runtime vs n_1

Figure 6.10 shows how long it takes to find both profile-based optimal matchings. The main observation is that both `Greedy-max-spa` and `Generous-max-spa` algorithms are scalable and can handle decent-sized instances in reasonable times.

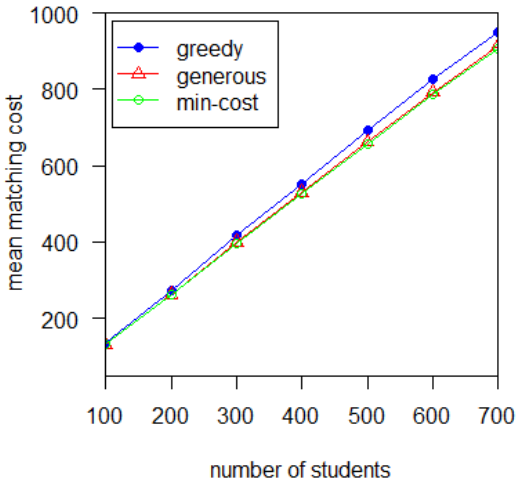

Figure 6.11: Mean matching cost vs n_1

Figure 6.11 shows how the cost of the matchings generated vary with the number of students. The cost seems to grow proportionally with the number of students. We observe that greedy maximum matchings have larger costs than generous and minimum cost maximum matchings. This corresponds to the mean degree curves shown in Figures 6.9 where greedy maximum matchings tend to match some students to projects further down their preference list thus adding to the cost of the matching. The average size of the matchings produced was very close to n_1 for all values of n_1 tested.

6.6.5.3 Varying preference list length

The length of students' preference list is one property that can be varied easily in practice (in the SPA context, it is often feasible to ask students to rank more projects if required). So, will increasing the length of the preference lists affect the quality of the matchings produced or the time taken to find them? For each value of R in the range $R \in \{1, 2, 3, ..., 10\}$ we tested this by varying the preference list lengths of $10,000$ randomly generated SPA instances. Each instance had $n_1 = 100$ students (with n_2, n_3, $C_{\mathcal{P}}$ and $C_{\mathcal{L}}$ all assigned their default values).

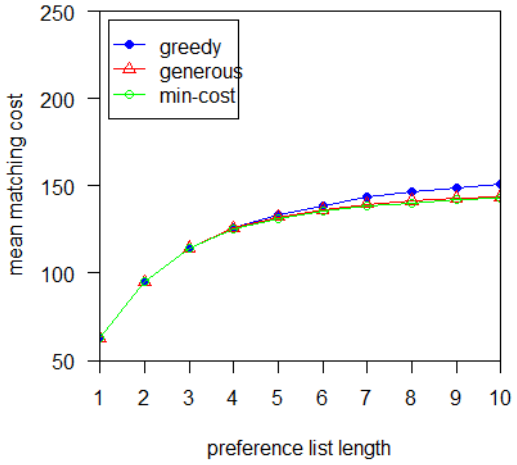

Figure 6.12: Mean matching cost vs R

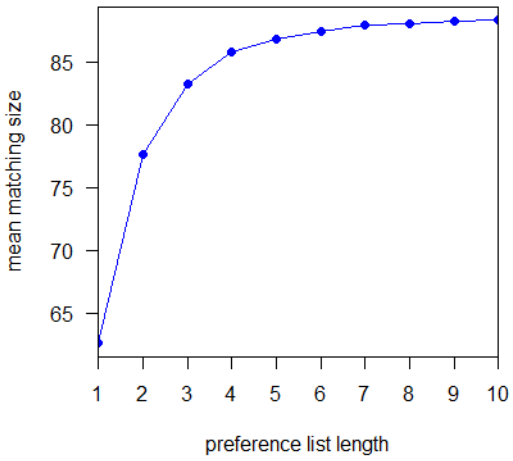

Figure 6.13: Mean matching size vs R

Figure 6.12 shows how the mean cost of the matchings obtained varied as we increased the preference list lengths. For the profile-based optimal matchings, the mean cost rises steeply from $R = 1$ to $R = 4$ but seems to level off beyond that. We observe that the overall cost of the matchings produced does not significantly change for $R > 5$. Thus asking students to submit preference lists greater than 5 will not significantly affect the overall quality of the generous and minimum cost maximum matchings obtained. Once again we observe a difference between the cost of the greedy maximum matchings and the other two.

Figure 6.13 also shows an important trend as it highlights the value of R beyond which there is little increase in the mean matching size of profile-based optimal matchings. For the instances generated in this experiment, that value is $R = 5$. Thus asking students to submit preference lists of length greater than 5 will not significantly affect the overall size of maximum matchings obtained. Figure 6.14 shows how the mean degree of the matchings varied as we increased preference list length. For values of $R \leq 3$ all matchings have the same mean degree as it is likely that some student gets her 3rd choice in each of these matchings. The curve for minimum cost maximum matchings is closer (with respect to degree) to that of generous maximum matchings

(obviously generous maximum matchings have lower degrees in general). They both seem to rise steeply for $R \leq 5$ and then level off at $R = 7$ and beyond. Thus asking students to submit preference lists greater than 7 will not significantly affect the overall degree of generous and minimum cost maximum matchings obtained. As expected, greedy maximum matchings had the highest degrees. For $R > 5$, the mean degree for greedy maximum matchings does not level off but continues to grow fairly steeply.

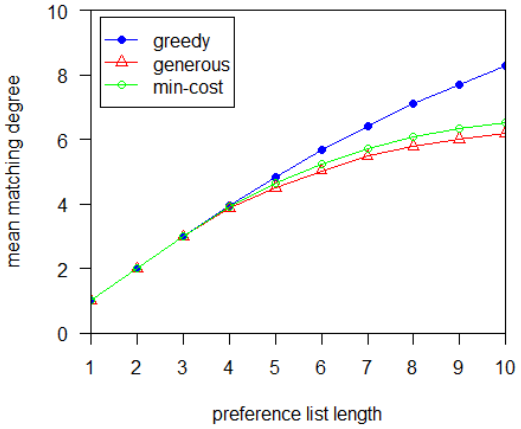

Figure 6.14: Mean matching degree vs R

Finally we consider how long it takes for the implemented algorithms to find their solutions. In general, the algorithms all seem to handle SPA instances with relatively long preference lists ($R = 10$) in reasonable time ($< 1.5s$).

6.6.5.4 Varying project popularity

Not all projects will be equally popular and so it is worth investigating the effects the relative popularity λ of the projects may have on the size and quality of the matchings produced. For these experiments, we set $n_1 = 100$ (with all the other default values) and varied the popularity of the projects involved from 0 to 9 in steps of 1, generating $1,000$ random instances for each popularity value. From Figure 6.15 we see that the cost of the matchings produced gradually increases as we increase the popularity ratio with the cost of the greedy maximum matching being slightly higher than the others (in line with other observations). From Figure 6.16 we observe no clear trend in the size of the matchings produced as we vary the popularity ratio.

Figure 6.17 shows the gaps between the mean degree of matchings produced using the various algorithms. Once again we see the mean degrees for the minimum cost and generous maximum matchings being considerably lower than that of the generous maximum matchings as the popularity ratio increases. Runtimes for the `Greedy-max-spa` and `Generous-max-spa` algorithms were less than $0.25s$.

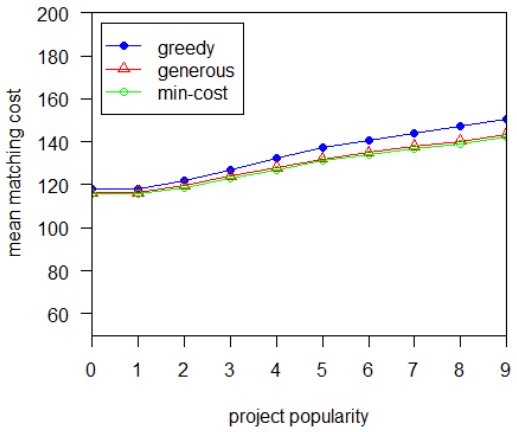

Figure 6.15: Mean matching cost vs popularity

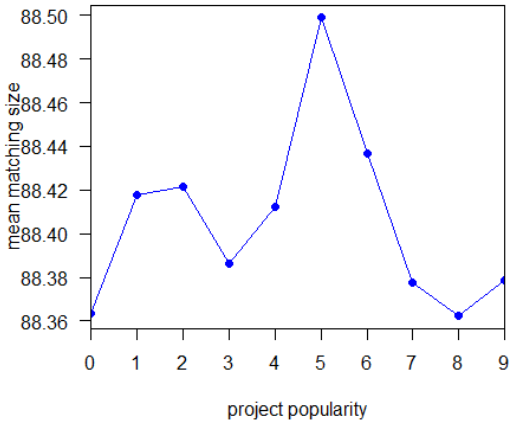

Figure 6.16: Mean matching size vs popularity

Figure 6.17: Mean matching degree vs popularity

6.6.6 Concluding remarks

Table 6.3 gives a breakdown of the profiles of the matchings obtained from 10,000 randomly generated SPA instances of size $n_1 = 100$ with preference list length $R = 10$. The mean size of the matchings found was 88.44 and therefore the percentages of students obtaining their ith choice ($1 \leq i \leq R$) sum up to 88.44 for each type of matching. The table shows the percentage of students with their first choice projects, second choice projects, and so on. Although the choice of which profile-based optimal matching is best will, in practice, be problem-specific, the results (as presented in Sections 6.6.4 and 6.6.5) give us a general idea of the strengths and weaknesses of the various optimality criteria. We summarise these points below.

With greedy maximum matchings we increase the percentage of students that are *happy* with their assigned projects (i.e., obtain their first choice). A rough estimate of how much better a greedy maximum matching is compared with other profile-based optimal matchings is the difference in the number of first-choice projects. Table 6.3 shows that

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Cost
Greedy:	64.59	13.29	4.86	2.23	1.28	0.81	0.54	0.37	0.26	0.20	137.06
Generous:	58.62	20.84	6.04	1.84	0.69	0.25	0.10	0.03	0.01	0.00	131.86
Minimum Cost:	61.19	17.72	6.08	2.09	0.83	0.33	0.13	0.05	0.02	0.01	130.86

Table 6.3: Mean matching profile and cost

the percentage of students with their first-choice project is higher when compared with minimum cost maximum matchings (by 3.40%) and even higher when compared with generous maximum matchings (by 5.97%). However this is achieved at the risk of also increasing the percentage of students who are *disappointed* with their assigned projects (we say a student s_i is disappointed with $p_j = M(s_i)$ if s_i is unmatched in M or $rank(s_i, p_j) > \lceil R/3 \rceil$).

With generous maximum matchings we reduce the percentage of disappointed students. Table 6.3 shows a significant improvement in the disappointment as we move from greedy maximum matchings (with 3.46% of matched students having their 5th choice or worse) to generous maximum matchings (with 1.08% of matched students having their 5th choice or worse). Although minimising disappointment among students is usually a very attractive property, this is achieved without considering the percentage of students who are happy with their assignments. Interestingly the generous criteria will continue to attempt to minimise the number of students matched to their nth choice project even as n tends to 1. This motivates a slightly more intuitive criteria where we initially adopt the generous criteria and, at some point (say at $n = 3$), switch to the greedy criteria.

Often the profile of a minimum cost maximum matching lies "in between" the two extremes given by a greedy maximum and generous maximum matching. This can be seen in terms of both the percentage of students with first-choice projects and those with fifth-choice projects or worse. In terms of the percentage of students with first-choice projects, the results show that minimum cost maximum matchings lie almost in the middle of the greedy and generous maximum matching percentages. In terms of those with fifth-choice projects or worse, it seems minimum cost maximum matchings are a lot closer to generous than greedy maximum matchings. This is usually seen as a desirable property.

6.7 Conclusion

In this chapter we defined what we consider to be the *basic* SPA model in which we impose only upper quotas on all three sets of agents (students, projects and lecturers).

We showed how greedy and generous maximum matchings can be found efficiently using network flow techniques. We also presented IP formulations for this basic SPA model and presented a range of empirical results obtained from evaluating them and the efficient algorithms. An obvious question to ask at this stage relates to which other extensions of SPA of practical relevance or theoretical significance can be investigated. We go some way in answering this question in Chapter 7. These include:

1. Can we improve on the $O(n_1^2 R m_2)$ algorithm for finding greedy and generous maximum matchings in SPA? One approach would be to determine whether there are faster ways of finding maximum profile augmenting paths in the underlying network than that presented in Algorithm 6.2. Another approach may be perhaps to abandon the network flow method and consider adopting other techniques used for solving similar problems in the CHAT (Capacitated House Allocation problem with Ties) context [54, 93, 48].

2. The notion of *Pareto optimality* has been well studied in the HA context [3, 7]. It is easy to see that the profile-based optimality criteria defined here imply Pareto optimality. However studying Pareto optimality in its own right is of theoretical interest. Since Pareto optimal matchings in CHAT can be of varying sizes, this extends to SPA. Given a SPA instance we may seek to find a maximum Pareto optimal matching in time faster than $O(n_1^2 R m_2)$.

Chapter 7

Further Algorithmic Results for SPA and its Variants

7.1 Introduction

Certain applications may require more constraints to be added to the basic SPA model defined in Chapter 6. An obvious example is the requirement that matchings should meet specified lower quotas of the agents involved. One of the main advantages of adopting the network flow approach is the flexibility it provides when considering these extra constraints. The idea remains the same - we model the SPA problems (including any extra constraints) as a network flow problems and use network flow algorithms (including the ones presented in Chapter 6) to solve them. In this chapter we investigate some extensions to SPA that we consider to be natural and show how our network flow models can be used to solve them.

In Section 7.2 we consider the problem of balancing the project supervision workload among the lecturing staff. This typically means that the task of supervising projects are evenly distributed among lecturers. We identify the addition of lower quotas as one possible means of achieving this and we present a network flow approach (using techniques presented in Chapter 6) to solve the problem. Due to potential infeasibility issues caused by the addition of lower quotas, we identify another means of balancing lecturer workload which involves defining a new optimality criterion. This involves minimising the difference between a lecturer with the most work and one with the least work. We present efficient algorithms for solving this problem. In Section 7.3 we consider the case where projects can only be feasible if a minimum number of students are assigned to them. We argue that simply assigning lower quotas (and so introducing infeasibility) is not ideal in this case. Thus we allow projects that do not meet their lower quotas to be abandoned (or closed). We show that finding a maximum

matching in this case is NP-hard. In view of this, we present two heuristics for finding large matchings which contain good profiles with respect to the profile-based optimality criteria. In Section 7.4 we extend the IP model presented earlier in Chapter 6 to handle SPA with project lower quotas where projects can be closed. The matchings produced are optimal with respect to size and profile. In Section 7.5 we present more empirical results relating to all the algorithms and techniques discussed in this chapter. We conclude the chapter in Section 7.6 with a list of interesting open problems.

7.2 Lecturer lower quotas and load balancing

7.2.1 Introduction

In SPA problems it is often required that the workload of supervising student projects is evenly spread across the lecturing staff (i.e., that project allocations are *load-balanced* with respect to lecturers). This is important because any project allocation should be seen by lecturers to be fair. Moreover a lecturer's workload may have an effect on her performance in other academic and administrative duties. In this section we identify various ways of measuring and achieving a load-balanced matching. This can be achieved by modifying the feasibility requirements or optimisation objectives (or both) of SPA. In Chapter 6 we identified the size and profile of a matching as the first and second optimisation objectives respectively. Typically the load-balancing objective would come in third position in this objective hierarchy.

7.2.2 Lecturer lower quotas

One way of achieving some notion of load-balancing with respect to lecturers is to introduce lower quotas. A lower quota on lecturer l_k is the minimum number of students that must be assigned to l_k in any feasible solution. We call this extension the *Student/Project Allocation problem with Lecturer lower quotas* (SPA-L). In an instance I of SPA-L, each lecturer l_k has an upper quota $d_k(I)^+ = d_k^+$ and now additionally has a lower quota $d_k^-(I)$ (it will be helpful to indicate specific instances to which these lower quotas refer within the notation). We assume that $d_k^-(I) \geq 0$ and $d_k^+(I) \geq \max\{d_k^-(I), 1\}$. In the SPA-L context, our definition of a matching as presented in Section 6.2 needs to be tightened slightly. A *constrained matching* is a matching M in the SPA context with the additional property that, for each lecturer l_k, $|M(l_k)| \geq d_k^-(I)$. A *constrained maximum matching* is a maximum matching taken over the set of constrained matchings in I. Suppose that L is the sum of the lecturer lower quotas in I (i.e. $L = \sum_{l_k \in \mathcal{L}} d_k^-(I)$)

students' preferences:	lecturers' offerings:
$s_1 : p_1 \quad p_2$	$l_1 : \{p_1\}$
$s_2 : p_3 \quad p_2$	$l_2 : \{p_2\}$
$s_3 : p_3$	$l_3 : \{p_3\}$

$c_1 = c_3 = 1$, $d_1^+ = d_3^+ = 1$ and $c_2 = d_2^+ = 2$
$d_1^- = d_3^- = 0$ and $d_2^- = 2$
Figure 7.1: A SPA-L instance I

and η is the size of a maximum matching in I[1]. For some k in $(L \leq k \leq \eta)$, let $\mathcal{M}'_k$ denote the set of constrained matchings of size k in I. A matching $M \in \mathcal{M}'_k$ is a *constrained greedy k-matching* if M has lexicographically maximum profile, taken over all matchings in $\mathcal{M}'_k$. An analogous definition for a *constrained generous k-matching* can be made.

Due to the introduction of these lecturer lower quotas, instances of SPA-L are not guaranteed to admit a feasible solution. Thus given an instance I of SPA-L, we seek to find a constrained greedy or a constrained generous maximum matching should one exist. We therefore present results analogous to Lemmas 6.3.1, 6.3.2 and 6.3.3. Firstly however, we make the following observations.

Proposition 7.2.1. *Given an* SPA-L *instance I, the size of a constrained maximum matching (should one exist) in I is equal to the size of a maximum matching in the underlying* SPA *instance in I.*

Proof. Assume I admits a constrained matching. Then, by dropping the upper quota of each lecturer $l_q \in \mathcal{L}$ from $d_q^+(I)$ to $d_q^-(I)$, and finding a saturating flow in the network obtained from the resulting instance, we can obtain a matching M_k of size k where $k = \sum_{l_q \in \mathcal{L}} d_q^-$. By returning the lecturer upper quotas to their original values and then successively finding and satisfying standard augmenting paths (starting from $f(M_k)$) we are bound to obtain a constrained maximum matching as lecturers do not lose any assigned students in the process. The absence of an augmenting path relative to the final flow is proof that the flow (and resulting constrained matching) is maximum. □

We also observe that a constrained greedy k-matching M_k in I need not be a greedy k-matching in I. That is, there may exist a matching M'_k of size k in I such that M'_k violates some of its lecturer lower quotas (i.e. M'_k is not a constrained matching) and $\rho(M'_k) \succ_L \rho(M_k)$. Figure 7.1 shows a SPA-L instance whose unique constrained greedy maximum matching is $M = \{(s_1, p_2), (s_2, p_2), (s_3, p_3)\}$ and a greedy maximum matching $M' = \{(s_1, p_1), (s_2, p_2), (s_3, p_3)\}$ such that $\rho(M') \succ_L \rho(M)$. However it is sufficient

[1]We will prove that η is equal to the size of a maximum constrained matching in Proposition 7.2.1.

to show that, starting from M_k, we can successively identify and augment (w.r.t. the incumbent flow) maximum profile augmenting paths in $N(I)$ until a constrained greedy maximum matching is found. Next we show that such augmenting paths exist.

Lemma 7.2.2. *Let I be an instance of* SPA-L *and let η denote the size of a constrained maximum matching in I. Let k $(1 \leq k < \eta)$ be given and suppose that M_k is a constrained greedy k-matching in I. Let $N = N(I)$ and $f = f(M_k)$. Then there exists an augmenting path P with respect to f in N such that if f' is the result of augmenting f along P then $M_{k+1} = M(f')$ is a constrained greedy $(k+1)$-matching in I.*

Proof. The proof is analogous to that presented for Lemma 6.3.1. We show that considering constrained matchings does not affect most of the arguments presented in the proof of Lemma 6.3.1. We will deal with the cases where considering constrained matchings may affect the arguments presented in the proof of Lemma 6.3.1. Firstly we observe that after cloning the projects in I to form a SPA-L instance $I' = C(I)$, the process of converting matchings in I to I' and vice versa is unaffected when the matchings considered are constrained. Thus since M_k is a constrained greedy k-matching in I, $C(M_k)$ is a constrained greedy k-matching in I'.

Let M'_{k+1} be a constrained greedy $(k+1)$-matching in I (this exists because $k < \eta$). Then $C(M'_{k+1})$ is a constrained greedy $(k+1)$-matching in I'. Let $X = C(M_k) \oplus C(M'_{k+1})$. Then each connected component of X is either (i) an alternating cycle, (ii)(a) an even-length alternating path whose end vertices are students, (ii)(b) an even-length alternating path whose end vertices are projects, (iii)(a) an odd-length alternating path whose end edges are in $C(M_k)$ or (iii)(b) an odd-length alternating path whose end edges are in $C(M'_{k+1})$. We firstly show that the procedures used to "join" and "eliminate" these connected components in Lemma 6.3.1 are unaffected when $C(M_k)$ and $C(M'_{k+1})$ are constrained matchings. The even-length components that we firstly consider are:

1. type (i) and type (ii)(a) alternating paths.
2. compound type (ii)(a) paths.

When considering the elimination of these even-length components (or compound paths), the requirement that the upper quotas of the lecturers involved must not be violated still holds even if the matchings considered are constrained. Moreover the number of students assigned to each lecturer never drops when considering the elimination of these even-length components (or compound paths).

Let $C(M''_{k+1})$ be the constrained greedy $(k+1)$-matching obtained from augmenting $C(M'_{k+1})$ along all these even-length paths. Then $X' = C(M_k) \oplus C(M''_{k+1})$ consists

of a set of compound type(ii)(b) paths, compound type (iii)(a) and compound type (iii)(b) paths. These paths, if considered independently, may lead to some lecturer losing an assigned student when they are used to augment $C(M_k)$ or $C(M''_{k+1})$. Thus the elimination argument, as presented in the proof of Lemma 6.3.1, does not hold. We modify this argument slightly as follows in the case of constrained matchings.

We firstly observe that $C(M_k)$ and $C(M''_{k+1})$ are constrained matchings. Thus augmenting $C(M_k)$ or $C(M''_{k+1})$ along X' leads to a constrained matching. When all the elements in X' are considered together, no lecturer violates her lower quota. If some lecturer loses a student due to some component of X' and drops below her lower quota, the she must gain an extra student due to another component in X'. But since $|C(M''_{k+1})|= |C(M_k)|+1$ there are q compound type (iii)(a) paths and $(q+1)$ compound type (iii)(b) paths in X' for some integer q. Compound type (ii)(b) components do not affect the size of the matchings.

We claim that there exists some compound type (iii)(b) path P' in X' such that when considering all the other components in X' (i.e. $X'\backslash P'$), lecturer upper and lower quotas are not violated and the size of the matchings are unchanged. Thus the elimination arguments presented in the proof of Lemma 6.3.1 can be applied to $X'\backslash P'$. P' can be extended to end with edge (l_p, v_t) such that $|M''_{k+1}(l_p)|> |M_k(l_p)|\geq d_p^-$. If $C(M'''_{k+1})$ is the constrained greedy $(k+1)$-matching obtained from augmenting $C(M''_{k+1})$ along $X'\backslash P'$, then $C(M'''_{k+1}) \oplus C(M_k) = P'$. If such a path P' does not exist then $|M''_{k+1}(l_p)|\leq |M_k(l_p)|$ for all $l_p \in \mathcal{L}$, a contradiction.

The rest of the proof for Lemma 6.3.1, involving the generation of an augmenting path, follows through. □

Lemma 7.2.3. *Let f be a flow in N and let $M_k = M(f)$. Suppose that M_k is a constrained greedy k-matching. Let P be a maximum profile augmenting path with respect to f. Let f' be the flow obtained by augmenting f along P. Now let $M_{k+1} = M(f')$. Then M_{k+1} is a constrained greedy $(k+1)$-matching.*

Proof. The proof for Lemma 6.3.2 holds even if $M(f)$ and $M(f')$ are constrained matchings as the number of students assigned to a lecturer never reduces as we augment f along P. □

Lemma 7.2.4. *Given an* SPA-L *instance I, let f be a flow in $N(I)$ where $k = |f|$ is not the size of a constrained maximum matching in I and $M(f)$ is a constrained greedy k-matching in I. Algorithm* `Get-max-aug` *finds a maximum profile augmenting path in $N(I)$ with respect to f.*

Proof. We observe that the proof presented for Lemma 6.3.3 also holds in this case even if $M(f)$ is a constrained greedy k-matching.

Algorithm 7.1 `Greedy-max-spa-l`

Require: SPA-L instance I;
Ensure: return a matching M if one exists or **null** otherwise;
1: copy I to from new instance I';
2: **for each** lecturer $l_k \in I'$ **do**
3: set $d_k^+(I') = d_k^-(I)$;
4: set $d_k^-(I') = 0$;
5: $\{I'$ becomes a SPA instance$\}$
6: $M' =$ `Greedy-max-spa`(I');
7: **if** $|M'| = \sum_{l_k \in \mathcal{L}} d_k^-(I)$ **then**
8: copy $f(M')$ in $N(I')$ into f in $N(I)$;
9: **loop**
10: $P =$ `Get-max-aug`$(N(I), f)$;
11: **if** $P \neq null$ **then**
12: augment f along P;
13: **else**
14: return $M(f)$;
15: **else**
16: return **null**;

The first part of the proof shows that after q iterations of the main loop of `Get-max-aug` where $0 \leq q \leq k$ the following is true: For every project $p_j \in \mathcal{P}$, $\rho(p_j) \succeq_L \Phi_{2q+1,2q}(p_j)$ where $\Phi_{2q+1,2q}(p_j)$ is the maximum profile of any partial augmenting path of length $\leq (2q+1, 2q)$ from an exposed student to p_j. By inspection, we observe that this argument remains unchanged even if $M(f)$ is a constrained matching in I.

The second part of the proof shows that a partial augmenting path P' (and subsequently a full augmenting path) can be constructed by following the *pred* values of projects and lecturers and the matched edges in $M(f)$ starting from some exposed project p_j with the maximum $\rho(p_j)$ profile going through some exposed student and ending at the source v_s. That is, we show that such a path is continuous and contains no cycle. We prove this by demonstrating that, should a cycle C exist, then augmenting f along C would yield a flow of the same size f' such that $M(f') \succ_L M(f)$ which is a contradiction to the fact that $M(f)$ is a greedy $k-$matching. This result also holds in the case where $M(f)$ is a constrained matching as any cycle found will not cause a lecturer to lose any assigned students and so the above arguments can still be made. □

Given Lemmas 7.2.2, 7.2.3 and 7.2.4, the `Greedy-max-spa` algorithm can be employed as part of an algorithm to find a constrained greedy maximum matching in a SPA-L instance should one exist. This new algorithm (which we call `Greedy-max-spa-l`) is presented in Algorithm 7.1. The algorithm takes an SPA-L instance I as input and returns a constrained greedy maximum matching M, should one exist, or **null** otherwise. A SPA instance I' is constructed from I by setting $d_k^-(I') = 0$ and $d_k^+(I') = d_k^-(I)$ for each lecturer l_k. Next we find a greedy maximum matching M' in I' using the

`Greedy-max-spa` algorithm. If $f' = f(M')$ is not a saturating flow (i.e., one in which all edges $(l_k, v_t) \in E_4$ are saturated), then I admits no constrained matching and we return **null**. Otherwise we augment flow f in $N(I)$ by calling the `Get-max-aug` algorithm, where f is the flow in $N(I)$ obtained from cloning f' in $N(I')$. We continuously augment the flow until no augmenting path exists. The matching $M = M(f)$ obtained from the resulting flow f is a greedy maximum constrained matching in I. Constrained generous maximum matchings can also be found in a similar way. We conclude with the following theorem.

Theorem 7.2.5. *Given a* SPA-L *instance I, a constrained greedy maximum matching and a constrained generous maximum matching in I can be obtained, should one exist, in $O(n_1^2 R m_2)$ time.*

Proof. Firstly we show that the matching M' obtained in Line 6 of the `Greedy-max-spa-l` algorithm is a constrained greedy $|f|$-matching in I. For contradiction, suppose that some other constrained matching M'' of the same size exists in I such that $\rho(M'') \succ_L \rho(M')$. Then since $|f| = \sum_{l_k \in \mathcal{L}} d_k^-(I)$, every lecturer has exactly the same number of assigned students in M' and M'', so M'' is a valid matching in I'. This contradicts the fact that M' is a greedy maximum matching in I'.

Lemmas 7.2.2, 7.2.3 and 7.2.4 prove that once we obtain a constrained greedy $|f|$-matching in I (should one exist), the rest of the algorithm finds a maximum constrained greedy maximum matching in I.

For finding a constrained generous maximum matching we simply replace the call to `Greedy-max-spa` in Line 6 and the call to `Get-max-aug` in Line 10 of the `Greedy-max-spa-l` algorithm with a call to the `Generous-max-spa` and the `Get-min-aug` algorithms respectively as described in Section 6.4. □

7.2.3 Minimising matching span

As demonstrated above, introducing lecturer lower quotas as part of the feasibility criteria is one way of achieving some notion of load-balancing. Another way is to define a load-balancing optimisation objective. Given a matching M in a SPA instance I, we define the *span* of M (denoted by $span(M)$) to be the difference between the maximum and minimum number of students assigned to a lecturer in M. That is, $span(M) = \max\{|M(l_k)|: l_k \in \mathcal{L}\} - \min\{|M(l_k)|: l_k \in \mathcal{L}\}$. The load-balancing objective is then to minimise $span(M)$ for all M in the set of greedy/generous maximum matchings in I.

One approach to solving the problem is to temporarily introduce upper and lower quotas and determine if a feasible solution to the resulting SPA-L instance exists. The

intuition is to firstly determine the size S^+ of a maximum matching and then to constrain the instance such that $span(M) = 0$ for all feasible matchings and attempt to find one such that $|M| = S^+$. If no such matching is found, we relax the constraints to allow $span(M) = 1$ and try again. We continue in this way until we find a maximum matching with the minimum value for $span(M)$. With the minimum value of $span(M)$ obtained (say $span(M) = x$), we apply the appropriate upper and lower quota constraints on the instance and then find a greedy maximum matching using the technique described in Section 7.2.2 above. In the worst case $span(M) = \max\{d_k^+(I) : l_k \in \mathcal{L}\}$ with the lecturer with the highest upper quota being full and some other lecturer being assigned no students.

In order to search for a greedy maximum matching M in I such that $span(M) \leq x$ for some x ($0 \leq x \leq \max\{d_k^+(I) : l_k \in \mathcal{L}\}$), we construct a constrained instance I' and set the upper quotas $d_k^+(I')$ and lower quotas $d_k^-(I')$ for all the lecturers l_k in I' as follows. We set $d_k^-(I') = \min\{d_k^+(I), y\}$ and $d_k^+(I') = \min\{d_k^+(I), d_k^-(I') + x\}$ for increasing values of y in the range $\lceil S^+/n_3 \rceil - x, \lceil S^+/n_3 \rceil - x + 1, \lceil S^+/n_3 \rceil - x + 2, ..., \lceil S^+/n_3 \rceil$ where n_3 is the number of lecturers in the instance. The intuition is to use our knowledge of the desired size of the matching to impose a restriction on the range of values of y that need to be considered, as not all values of y will yield a matching of size η. Setting lower and upper quotas based on this reduced range helps minimise the amount of unnecessary work. Algorithm 7.2 describes the process. We start with an initial target span of $x = 0$ and with $d_k^-(I') = \min\{d_k^+(I), \lceil S^+/n_3 \rceil - x\}$ and seek to find a feasible constrained matching in I' such that $|M| = S^+$. If no such matching is found, we move on to the next value of $d_k^-(I')$ (and thus $d_k^+(I')$). If no feasible constrained matching is found for all values of $d_k^-(I')$ in the range, we relax our target span by incrementing x by 1 and starting again. The process continues until a greedy maximum matching is found.

Lemma 7.2.6. `Alg-min-span` *finds a greedy maximum matching of minimum span in* $O(n_1^2 nm)$ *time.*

Proof. Let M be a greedy maximum matching in I and let $S^+ = |M|$ and $span(M) = x$ for some value of x ($0 \leq x \leq \max\{d_k^+(I) : l_k \in \mathcal{L}\}$). We show that the range of values considered for $d_k^-(I')$ (and thus $d_k^+(I')$) is necessary and sufficient for finding M. Firstly we observe that the values of $d_k^+(I')$ ensures that $span(M) \leq x$ for all values of x and $d_k^-(I')$ considered, so we need only focus on the range of values of $d_k^-(I')$.

We claim that $\min\{d_k^+(I), \lceil S^+/n_3 \rceil - x\} \leq \min\{|M(l_k)|: l_k \in \mathcal{L}\} \leq \min\{d_k^+(I), \lceil S^+/n_3 \rceil\}$. Suppose not and $\min\{|M(l_k)|: l_k \in \mathcal{L}\} < \lceil S^+/n_3 \rceil - x$. There exists some lecturer l_k such that $|M(l_k)| = \lceil S^+/n_3 \rceil - (x+1)$. Thus every lecturer can have at most $\lceil S^+/n_3 \rceil - 1$ assigned students in order to ensure $span(M) \leq x$. Thus $|M| \leq n_3(\lceil S^+/n_3 \rceil - 1)$. Since

Algorithm 7.2 `Alg-min-span`

Require: SPA instance I;
Ensure: return a greedy maximum matching M of minimum span;
1: $x := 0$;
2: $S^+ :=$ size of a maximum matching in I;
3: **while** $x \leq \max\{d_k^+(I) : l_k \in \mathcal{L}\}$ **do**
4: $y := \min\{0, \lceil S^+/n_3 \rceil - x\}$;
5: **while** $y \leq \lceil S^+/n_3 \rceil$ **do**
6: generate SPA-L instance I' from I;
7: set $d_k^-(I') := \min\{d_k^+(I), y\}$ for each lecturer $l_k \in \mathcal{L}$;
8: set $d_k^+(I') := \min\{d_k^+(I), d_k^-(I') + x\}$ for each lecturer $l_k \in \mathcal{L}$;
9: $M :=$ maximum constrained matching in I';
10: **if** $M \neq$ **null** and $|M| = S^+$ **then**
11: $M :=$ greedy maximum constrained matching in I';
12: return M;
13: **else**
14: $y := y + 1$;
15: $x := x + 1$;

$\lceil S^+/n_3 \rceil - 1 < S^+/n_3$, it follows that $|M| < S^+$ for all possible values of x, a contradiction to the fact that M is a maximum matching in I. For the second inequality, suppose $\min\{|M(l_k)|: l_k \in \mathcal{L}\} > \lceil S^+/n_3 \rceil$. Then every lecturer will have at least $\lceil S^+/n_3 \rceil + 1$ assigned students. Thus $|M| \geq n_3(\lceil S^+/n_3 \rceil + 1)$. Thus $|M| \geq S^+ + n_3 > S^+$ for all possible values of x, a contradiction to the fact that η is the size of a maximum matching in I.

Since the `Alg-min-span` algorithm considers x in increasing order from 0 to $\max\{d_k^+(I) : l_k \in \mathcal{L}\}$, it is guaranteed to find a greedy maximum matching of minimum span.

The algorithm calls a maximum flow algorithm $O(n_1^2)$ times in the worse case thus making the overall runtime of the algorithm $O(n_1^2 nm)$ assuming we use the $O(nm)$ max flow algorithm by Orlin [95] where n and m are the total number of nodes and edges in the underlying network. □

7.2.4 Minimising span with lower quotas

It is also feasible to model SPA problems to adopt both the feasibility criterion (by enforcing lecturer lower quotas) and the optimisation objective (of minimising lecturer span) presented in the previous subsection. Given an instance I of SPA-L, the span $span(M)$ of a feasible matching M in I is given by

$$span(M) = \max\{|M(l_k)| - d_k^-(I) : l_k \in \mathcal{L}\} - \min\{|M(l_k)| - d_k^-(I) : l_k \in \mathcal{L}\}$$

This is a measure of the maximum difference between the *excess* number of students assigned to any pair of lecturers in M. The formula reduces to the case presented in Section 7.2.3 above if the lower quotas are dropped (i.e. $d_k^-(I) := 0 : l_k \in \mathcal{L}\}$).

This problem can be solved by slightly modifying Algorithm 7.2 to consider lower quotas. We modify the lower and upper quota assignments (in Lines 7 and 8 respectively) as follows. For the lower quotas of each lecturer $l_k \in \mathcal{L}$, we set $d_k^-(I') := \min\{d_k^+(I), d_k^-(I)+y\}$ and for the upper quotas, we set $d_k^+(I') := \min\{d_k^+(I), d_k^-(I')+x\}$. The resulting algorithm will find a constrained greedy maximum matching of minimum span in an instance I of SPA-L should one exist.

7.3 Project lower quotas

7.3.1 Introduction

As in the case with lecturer lower quotas, SPA can also be extended to allow for project lower quotas. This is motivated by the idea that certain projects may require a minimum number of assigned students in order for them to proceed. We call this extension the *Student/Project problem with Project Lower quotas* (SPA-PL). In an instance I of SPA-PL, each project p_j has an upper quota $c_j^+ = c_j$ and now additionally has a lower quota c_j^-. We assume that $c_j^- \geq 0$ and $c_j^+ \geq \max\{c_j^-, 1\}$. In this context (as in SPA-L), our definition of a matching as presented in Section 6.2 needs to be tightened slightly. Two variants of the problem, which depend on our definition of a feasible matching, can be described. These variants are analogous to the two models for the *Hospitals/Residents problem with Lower Quotas* (HR-LQ) described in [85].

In the first variant of the problem, a matching M is required to satisfy all the constraints that M does in the SPA context with the additional property that, for each project p_j, either $|M(p_j)| \geq c_j^-$ or $|M(p_j)| = 0$. Thus if a project does not meet its lower quota, it is simply *closed* and will not feature at all in the resulting matching. Projects that do meet their lower quotas in a given matching M are said to be *open* in M. An alternative variant of the problem is one in which a matching M is required to satisfy all the constraints that M does in the SPA context with the additional property that, for each project p_j, $|M(p_j)| \geq c_j^-$. As in the case for lecturer lower quotas, this extra constraint implies that an instance need not admit a feasible solution.

For the rest of this chapter, we only consider the first variant of the problem. That is, we seek to find a greedy or generous maximum matching given an SPA-PL instance where projects may be closed.

7.3.2 Hardness of MAX SPA-PL

We consider the problem of finding greedy and generous matchings given an SPA-PL instance. Firstly we observe that matchings may be of varying sizes. For example, given a non-empty matching M, closing any project that is open in M would yield a feasible matching of a smaller size. We define MAX SPA-PL as the problem of finding a maximum matching given an instance of SPA-PL. In the corresponding decision problem denoted MAX SPA-PL-D, we ask whether there exists a matching of size at least K given an instance of SPA-PL where $0 < K \leq n_1$. We show that MAX SPA-PL-D is NP-complete. This results holds if, in addition, the matching is required to be a greedy or generous maximum matching. Special cases of the problem, which restrict the lengths of students' preference lists or the maximum value of the projects' lower and upper quotas, can also be defined. We denote (x,y)-MAX SPA-PL as the problem of finding a greedy maximum matching given an instance of SPA-PL, in which each student is allowed to rank at most x projects and y is the maximum upper quota of any project. We replace x or y with ∞ to indicate that the parameter in question is unbounded. (x, y)-MAX SPA PL-D is the decision version of (x, y)-MAX SPA PL obtained in an analogous fashion to MAX SPA PL-D.

Theorem 7.3.1. (2,3)-MAX SPA-PL-D *is NP-complete.*

Proof. We reduce from *Maximum Independent Set* (MIS) for cubic graphs which is defined as follows. Given a cubic graph G and an integer K, MIS is the problem of determining whether G contain an independent set of size at least K. MIS is NP-complete even for cubic graphs [83, 32]. Let $\langle G, K\rangle$ be an instance of MIS, where $G = (V, E)$, $E = \{e_1, \ldots, e_m\}$ and $V = \{v_1, \ldots, v_n\}$. Construct an instance J of SPA-PL as follows. Let $\mathcal{S} = \{s_1, \ldots, s_m\}$, let $\mathcal{P} = \{p_1, \ldots, p_n\}$ and let $\mathcal{L} = \{l_1, \ldots, l_n\}$. For each i $(1 \leq i \leq m)$, suppose that e_i is incident to v_r and v_s in G. Student s_i finds both p_r and p_s acceptable and is indifferent between them. For each j $(1 \leq j \leq m)$, project p_j satisfies $c_j^- = c_j^+ = 3$ and is offered by lecturer l_j, who satisfies $d_j = 3$.

We claim that G has an independent set of size at least K if and only if J admits a greedy/generous k-matching for $k \geq 3K$.

For, let $S = \{v_{i_1}, \ldots, v_{i_k}\}$ be an independent set in G, where $k \geq K$. Form a set of edges M in J as follows. For each r $(1 \leq r \leq k)$, let e_{j_1}, e_{j_2} and e_{j_3} be the three edges incident to v_{i_r}. Add (s_{j_t}, p_{i_r}) to M $(1 \leq t \leq 3)$. Then $|M| = 3k$, and moreover M is a matching in J since S is an independent set in $\langle G, K\rangle$, and each project has either 0 or 3 assignees in J. Thus J admits a greedy/generous $|M|$-matching where $|M| \geq 3K$.

Conversely let M be a matching in J of size $k' \geq 3K$. Form a set of vertices S in G as follows. For each j $(1 \leq j \leq m)$, either project p_j has 0 or 3 assignees. In the

latter case add v_j to S. Then $|S| = k'/3 \geq K$, and moreover S is an independent set in $\langle G, K \rangle$ since M is a matching in J. □

Corollary 7.3.2. *The problem of finding a greedy/generous maximum matching, given an instance of* $(2,3)$-MAX SPA PL *is NP-hard.*

When considering (x,y)-MAX SPA-PL, although y refers to the maximum upper quota of any project, by the reduction in the proof of Theorem 7.3.1 we can also consider y to mean the maximum lower quota of any project or the maximum number of students applying to any project and Theorem 7.3.1 will still hold in the case that $y = 3$. An obvious way of offsetting this hardness is to consider approximation algorithms for MAX SPA-PL. However Theorem 7.3.3 shows that the problem is hard to approximate to within a constant factor (this result is due to [23]).

Theorem 7.3.3 ([23])**.** *Given any* $k \geq 3$*, it is NP-hard to approximate* (∞, k)-MAX SPA-PL *to within a factor of* $k/\ln k$ *where* k *is the maximum number of students applying for any project.*

Proof. We reduce from the MAXIMUM k-SET PACKING (k-SP) problem which is defined as follows. Given a set of elements $\mathcal{E} = \{e_1, e_2, ..., e_{n_1}\}$ and a set $\mathcal{C} = \{C_1, C_2, ..., C_{n_2}\}$ of subsets of $\mathcal{E}$ with each subset $C_i \in \mathcal{C}$ having k elements, k-SP is the problem of finding the maximum number of pairwise disjoint sets from $\mathcal{C}$. It is NP-hard to approximate k-SP to within a factor of $k/\ln k$ [26].

Let I be an instance of k-SP. Construct an instance I' of MAX SPA-PL as follows. Let the set of students $\mathcal{S} = \{s_1, s_2, ..., s_{n_1}\}$ correspond to $\mathcal{E}$ and the set of projects $\mathcal{P} = \{p_1, p_2, ..., p_{n_2}\}$ correspond to $\mathcal{C}$. Given any j $(1 \leq j \leq n_2)$, for each element $e_i \in C_j$, we add project p_j to A_i (i.e. student s_i finds p_j acceptable). Each project p_j is offered by a single unique lecturer l_j and $c_j^- = c_j^+ = d_j = k$. Students are indifferent between acceptable projects thus any maximum matching is also a greedy maximum matching in I'.

Let $s^+(I)$ and $m^+(I')$ denote the size of a maximum set packing in I and a greedy maximum matching in I' respectively. Let $\mathcal{C}' \subseteq \mathcal{C}$ be a maximum set packing in I (i.e. $|\mathcal{C}'| = s^+(I)$). We can construct a matching M in I' as follows. For each $e_i \in C_j$ where $C_j \in \mathcal{C}'$ we add the pair (s_i, p_j) to M. It is easy to see that M is a valid matching in I' and, since $|C_j| = k \ \forall \ C_j \in \mathcal{C}$, $|M| = ks^+(I)$ and so $m^+(I') \geq ks^+(I')$.

Also let M be a greedy maximum matching in I' (i.e. $|M| = m^+(I')$). We construct a solution $\mathcal{C}' \subseteq \mathcal{C}$ in I' by adding the set C_j to $\mathcal{C}'$ for every project p_j where $|M(p_j)| > 0$. Given the values of c_j^- and c_j^+ in I', it is easy to see that $\mathcal{C}' \subseteq \mathcal{C}$ is a valid set packing in I and $|\mathcal{C}'| = m^+(I')/k$ and so $s^+(I') \geq m^+(I')/k$.

Thus $m^+(I') = ks^+(I')$ and an approximation algorithm for (∞, k)-MAX SPA-PL with performance guarantee better than $k/\ln k$ would approximate k-SP by the same factor, a contradiction. □

7.3.3 Heuristics for MAX SPA-PL

7.3.3.1 Introduction

In this subsection we present heuristics for MAX SPA-PL (given the NP-hardness of the problem). As stated above, in a given matching, each project must either meet its lower quota (be *open*) or have no student matched to it (be *closed*). We have shown above that the problem is NP-hard to solve or approximate. To deal with this hardness we can relax the requirement that the matching must be maximum in order to produce heuristics that find "good" matchings both in terms of size and profile. Although the final matchings obtained need not be optimal with respect to size, we seek to find matchings that are as large as possible and subject to that, we seek to optimise these matchings with respect to some profile-based optimality criterion.

Let I be a MAX SPA-PL instance. Also let I' be the underlying SPA instance (i.e. without considering lower quotas) and let M be a matching in I'. Since M may not be a feasible matching in I, we define $\mathcal{P}'(M) = \{p_j : M(p_j) < c_j^-(I)\}$ to be the set of *violating projects* that do not meet their lower quotas in M with respect to I. Projects in $\mathcal{P}'(M)$ can potentially receive more students in order to meet their lower quotas and become open, or they can give up their assigned students and become closed. Let $N = N(I')$ be the network derived from I'. Then $f = f(M)$ is a flow in N. A violating project $p_j \in \mathcal{P}'(M)$ can be closed by setting $f(s_i, p_j) = c(s_i, p_j) = 0$ for all students s_i such that $p_j \in A_i$, setting $f(v_s, s_i) = 0$ for all students $s_i \in M(p_j)$, setting $f(p_j, l_k) = c(p_j, l_k) = 0$ and setting $f(l_k, v_t) = f(l_k, v_t) - |M(p_j)|$ where lecturer $l_k = l(p_j)$. The resulting flow is thus reduced by $|M(p_j)|$. A previously closed project p_j can be *reactivated* by setting the capacities of its incident edges to their original values in N. Thus $c(s_i, p_j) = 1$ for all students s_i such that $p_j \in A_i$ and $c(p_j, l_k) = c_j^+(I)$ where $l_k = l(p_j)$.

The heuristics presented in Sections 7.3.3.2 and 7.3.3.3 involve steps in which projects may be closed or reactivated. Both heuristics start by finding a profile-based optimal matching M in I' that admits a set of violating projects $\mathcal{P}'(M)$ in I. Violating projects are then handled serially by either assigning more students to them or closing them. Both heuristics handle projects in $\mathcal{P}'(M)$ in different ways in order to arrive at a valid matching in I (one in which all projects are either open or closed). In the following

Algorithm 7.3 `Heuristic-cps`

Require: SPA-PL instance I;
Ensure: return matching M;
1: Generate I' from I by dropping project lower quotas;
2: $N := N(I')$;
3: **loop**
4: $M :=$ `Greedy-max-spa`(I');
5: $\mathcal{P}'(M) := \{p_j : |M(p_j)| < c_j^-(I)\}$;
6: **if** $|\mathcal{P}'(M)| > 0$ **then**
7: close some project $p_j \in \mathcal{P}'(M)$ in N;
8: **else**
9: return M;

sections we present both heuristics and in Section 7.5.4 we show results obtained from evaluating them empirically.

7.3.3.2 Close projects serially (HEURISTIC-CPS)

In this heuristic, violating projects are handled by closing them one after the other (in a serial manner). After each project is closed, the augmenting path algorithm (say `Greedy-max-spa`) is re-executed on the resulting instance. This process continues until there are no violating projects remaining.

Given a MAX SPA-PL instance I, we start by forming I' (i.e., relaxing the project lower quotas). We then find a greedy maximum matching M in I' by calling the `Greedy-max-spa` algorithm. Next we consider all the projects in $\mathcal{P}'(M)$ and begin to close them one after the other. After each project in $\mathcal{P}'(M)$ is closed, we re-execute the `Greedy-max-spa` algorithm on the resulting instance. Although this may cause other projects that were previously open to violate their lower quotas (thus adding to $\mathcal{P}'(M)$), it is expected that a greater number of violating projects will gain enough students to become open. Once a project is closed it is not reactivated and will remain closed in the final matching obtained. The process continues until $\mathcal{P}'(M)$ is empty (it must terminate because every time the main loop iterates, either we return a matching or we close exactly one project). Algorithm 7.3 describes the entire process.

Although projects are closed randomly in Algorithm 7.3, the order may affect the size of the final matching obtained. There are various ways of ordering the violating projects to be closed. One approach is to order violating projects based on the number of extra students required for them to achieve their lower quotas (i.e., $c_j^- - |M(p_j)|$). The higher the number of extra students required, the earlier the project is closed. This will allow projects with a greater potential of achieving their lower quotas to have more opportunities to do so. Another idea is to close projects that are less popular

as they are less likely to improve (and become open) during the process. If the most popular projects are given more opportunities (i.e., are not closed early on) they have a better chance of achieving their lower quotas. Finally we may order the violating projects based on the number of students they have assigned to them at that time (i.e., $|M(p_j)|$ for project p_j). The idea is that projects with a large number of students assigned to them will contribute more to the final matching than those with a smaller number of assigned students and so should not be closed early on. These orderings can be evaluated empirically in order to determine how well they perform in practice.

Since no previously closed projects are reactivated and the number of projects is finite, the algorithm is bound to terminate. In the worst case, all of the n_2 projects will need to be closed thus leading to a time complexity of $O(n_1^2 n_2 R m_2)$.

7.3.3.3 Open projects serially (`Heuristic-ops`)

In this heuristic all violating projects are initially closed. They are then reactivated one after the other (in a serial manner). Given a MAX SPA-PL instance I, as in the case of `Heuristic-cps`, we start by finding I' (i.e. relaxing the project lower quotas). We then find a greedy maximum matching M in I' by calling the `Greedy-max-spa` algorithm. Next, we close all the projects in $\mathcal{P}'(M)$ and begin to reactivate them one after the other. After each project in $\mathcal{P}'(M)$ is reactivated, we re-execute the `Greedy-max-spa` algorithm on the resulting instance. Although this may cause other projects that were previously open to violate their lower quotas (as $\mathcal{P}'(M)$ can change at each iteration with some projects leaving and others arriving), once again, it is expected that a greater number of violating projects will gain enough students to become open (i.e. leave $\mathcal{P}'(M)$). Each project can only be reactivated once. If a previously reactivated project remains or becomes a violating project, it is permanently closed and will remain closed in the resulting matching. Algorithm 7.4 details the entire process.

Once again the order in which projects are reactivated may affect the size of the final matching obtained. As suggested in the `Heuristic-cps` case, we may choose to reactivate popular projects first or projects needing the fewest number of extra students to meet their lower quotas. We may also choose to reactivate projects with a larger number of assigned students first. The algorithm is bound to terminate (as we reactivate a project at most once) and has a time complexity of $O(n_1^2 n_2 R m_2)$.

Algorithm 7.4 `Heuristic-ops`

Require: SPA-PL instance I;
Ensure: return matching M;
1: Generate I' from I by dropping project lower quotas;
2: $N := N(I')$;
3: $\mathcal{P}''(M) := \{\}$; /* store already reactivated projects */
4: **loop**
5: $M :=$ `Greedy-max-spa`(I');
6: $\mathcal{P}'(M) := \{p_j : |M(p_j)| < c_j^-(I)\}$;
7: close all projects $p_j \in \mathcal{P}'(M)$ in N;
8: **if** $|\mathcal{P}'(M) \backslash \mathcal{P}''(M)| > 0$ **then**
9: reactivate some project $p_j \in \mathcal{P}'(M) \backslash \mathcal{P}''(M)$ in N;
10: add p_j to $\mathcal{P}''(M)$;
11: **else**
12: return M;

7.4 Extending the SPA IP model

The IP models presented in Section 6.5 only enforce upper quota constraints on the projects and lecturers. Extra constraints can be added to the models in order to enforce lecturer and project lower quotas. However doing so may result in problem instances that admit no feasible solutions. The following constraints can be added to enforce lecturer lower quotas.

$$1. \qquad \sum_{p_j \in P_k} \sum_{s_i \in S_j} x_{i,j} \geq d_k^- \qquad \forall\, l_k \in \mathcal{L}$$

For project lower quotas, as identified in Section 7.3.1, two feasibility definitions exist. For the SPA-PL variant where all projects must meet their lower quotas, instances need not admit a feasible solution. An extra constraint (similar to Constraint 1 above) can be added to the model in order to enforce the project lower quotas.

$$2. \qquad \sum_{s_i \in S_j} x_{i,j} \geq c_j^- \qquad \forall\, p_j \in \mathcal{P}$$

For the SPA-PL variant where a project must either meet its lower quota or have no students assigned to it (i.e., become closed), the following constraints can be added to the model.

$$2a. \quad \sum_{s_i \in S_j} x_{i,j} + c_j^+ y_j \leq c_j^+ \qquad \forall\ p_j \in \mathcal{P}$$

$$2b. \quad \sum_{s_i \in S_j} x_{i,j} + c_j^- y_j \geq c_j^- \qquad \forall\ p_j \in \mathcal{P}$$

If a project p_j has no assignee, Constraint $2b$ is satisfied if $y = 1$, in which case Constraint $2a$ is satisfied. On the other hand if p_j has at least one assignee, Constraint $2a$ is satisfied if $y = 0$ and p_j meets its upper quota and then Constraint $2b$ is satisfied only if p_j meets its lower quota.

7.5 Empirical evaluation

7.5.1 Introduction

Some experimental results in the SPA context have already been presented in Section 6.6. In this section we present more results relating to experiments carried out on the implemented algorithms and heuristics presented in this Chapter. We implemented and evaluated the `Alg-min-span` algorithm for finding a maximum matching of minimum span in SPA and SPA-L instances. We also implemented the heuristics for finding large matchings in MAX SPA-PL instances.

Given that, in the case of MAX SPA-PL, the matchings produced are not guaranteed to be optimal with respect to size or profile, we adopt a measure of accessing the *profile quality* of the matching produced when compared with an optimal solution (obtained from the IP model). Following [21], we can define the *rank approximation* α of a matching M in the SPA context as follows. Matching M has rank approximation α if, for all positions r $(1 \leq r \leq R)$,

$$|\{(s_i, p_j) \in M : rank(s_i, p_j) \leq r\}| \ \geq \ \frac{1}{\alpha} |\{(s_i, p_j) \in M_{opt} : rank(s_i, p_j) \leq r\}|$$

where M_{opt} is an optimal (i.e., a greedy maximum) matching. We therefore use this measure to evaluate the quality of a matching produced by the MAX SPA-PL heuristics. Given a SPA instance that admits a non-empty matching M and a greedy maximum matching M_{opt}, the profile quality (w.r.t. the greedy criterion) $quality(M)$ is given as follows.

$$quality(M) = \max\left\{\frac{|\{(s_i,p_j) \in M_{opt} : rank(s_i,p_j) \leq r\}| \ + \ 1}{|\{(s_i,p_j) \in M : rank(s_i,p_j) \leq r\}| \ + \ 1} : 1 \leq r \leq R)\right\}$$

If M_{opt} is a generous maximum matching in I the profile quality (w.r.t. the generous criterion) $quality(M)$ is given as follows.

$$quality(M) = \frac{|M_{opt}|}{|M|} \times \max\left\{\frac{|\{(s_i,p_j) \in M : rank(s_i,p_j) \geq r\}| \ + \ 1}{|\{(s_i,p_j) \in M_{opt} : rank(s_i,p_j) \geq r\}| \ + \ 1} : 1 \leq r \leq R)\right\}$$

In both cases, the closer the quality is to 1, the better. Thus in the following experiments, we consider quality only in the greedy case. The input parameters used to generate random instances are the same as those presented in Section 6.6 unless otherwise stated.

7.5.2 Lecturer lower quotas

Due to the potential feasibility problems that may arise in the case of lecturer lower quotas, we investigated the percentage solvability of randomly generated SPA-L instances while increasing the ratio of the lower to upper quotas assigned to the lecturers. Results from these experiment would potentially give us an indication of levels of lecturer lower to upper quota ratios within which feasibility levels are deemed acceptable. The total number of lecturer lower quotas $L_{\mathcal{L}}$ was varied from 0 to $C_{\mathcal{L}}$ where $C_{\mathcal{L}}$ is the total number of lecturer upper quotas (i.e. $L_{\mathcal{L}}$ was varied from $0\%C_{\mathcal{L}}$ to $100\%C_{\mathcal{L}}$ by steps of 10%). The $L_{\mathcal{L}}$ and $C_{\mathcal{L}}$ values were evenly distributed among the lecturers involved.

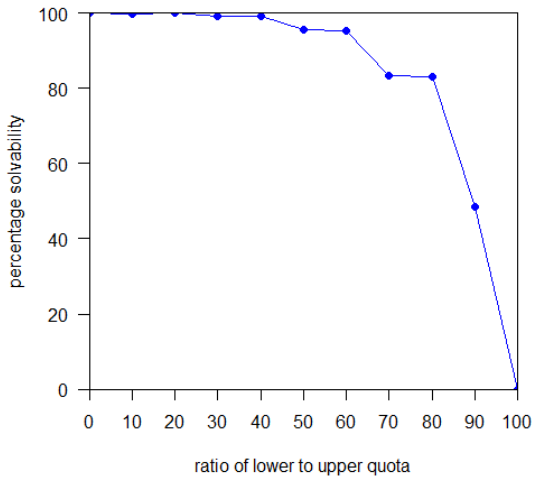

Figure 7.2: % solvability vs $L_{\mathcal{L}}/C_{\mathcal{L}}$

For each value of $L_{\mathcal{L}}$, 1000 instances of SPA PL were generated and solved using the

extended IP model. Figure 7.2 shows the results obtained with the percentage solvability dropping from 0% to 100% as we increase the total lecturer lower quotas from $0\%C_{\mathcal{L}}$ to $100\%C_{\mathcal{L}}$.The results suggest that for instances where $L_{\mathcal{L}}/C_{\mathcal{L}} \leq 0.6$, feasibility percentages are very high. However as the gap between upper and lower quotas is reduced beyond this point, the feasibility ratio drops steeply.

7.5.3 Minimising matching span

`Alg-min-span` was implemented and tested on real world datasets and randomly generated SPA instances. Table 7.1 shows the minimum span, the maximum possible span and the average lecturer allocation for a number of real-world SPA instances obtained from previous runs of the student-project allocation scheme at the School of Computing Science, University of Glasgow. With a maximum possible span of 3 for all the instances considered, the matchings produced had an average minimum span of 2. Thus on average, some lecturers get two more students than other lecturers. When considering the results in conjunction with the average allocations per lecturer, one may conclude that the minimum span values obtained are "poor". For example, in the 2013/2014 session we have one lecturer having more than double the average workload. Also none of the instances produced a matching where all the lecturers had the same number of students assigned to them. These results may be attributed to the variation in the popularity of the projects offered by lecturers and variations in the upper quotas of the lecturers. So a lecturer who offers popular projects may be more likely to end up with a larger number of assigned students.

Session	$\min\{span(M)\}$	$\max\{d_k^+\}$	$\text{avg}\{\lvert M(l_k)\rvert\}$
2014/2015	2	3	1.38
2013/2014	3	3	1.28
2012/2013	2	3	1.12
2011/2012	2	3	1.19
2010/2011	1	3	1.17
2009/2010	2	3	1.14

Table 7.1: Span of matchings in real-world SPA instances

We generated random SPA instances and ran the `Alg-min-span` algorithm on them. We measured the minimum span produced, the quality of the matchings as well as the maximum upper quota of any lecturer in each instance. For each instance generated, we also executed the `Greedy-max-spa` algorithm and measured the span of the resulting algorithms. The number of lecturers and the total upper quotas of all lecturers are given as $n_3 = 0.2n_1$ and $C_{\mathcal{L}} = 1.2n_1$ respectively.

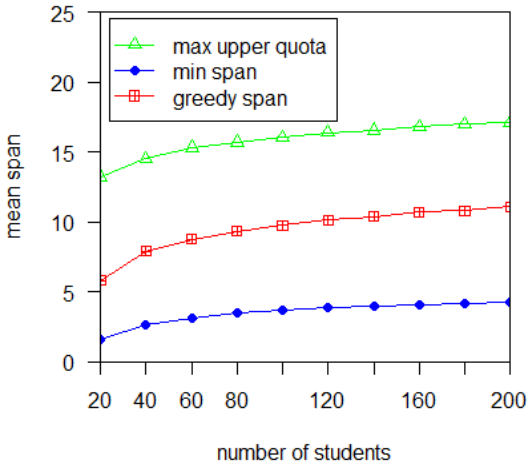

Figure 7.3: Minimum span vs n_1

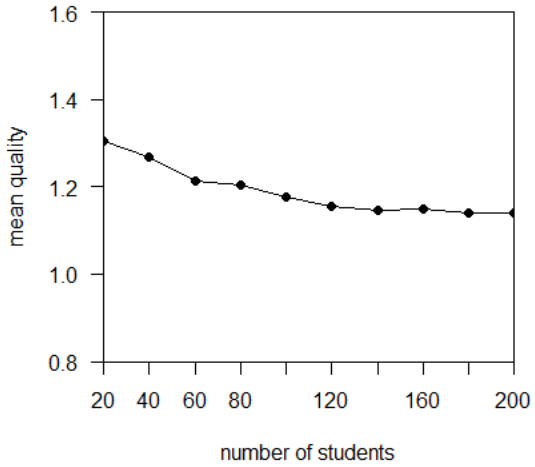

Figure 7.4: Mean quality vs n_1

Figure 7.3 shows the results obtained. We observe a considerable improvement in the span when compared with the worst-case scenario (i.e., with one lecturer with the highest upper quota being full and another having no assigned partner). What we do not see is any measurable change in this gap as we increase the instance size. This observation is further highlighted by the fact that the overall quality of the minimum span maximum matchings produced does not vary greatly (between 1.1 and 1.3) for all the instances measured (as shown in Figure 7.4). One possible explanation for this is the fact that the sub-routines used in the `Alg-min-span` algorithm attempt to find matchings that are as greedy as possible.

7.5.4 Evaluating Heuristics for MAX SPA-PL

So far we have evaluated SPA instances where projects and lecturers have no lower quotas. In this section we consider the case where projects have lower quotas. We restrict our analysis to the SPA-PL model and evaluate the performance of both heuristics described in Section 7.3.3. Both `Heuristic-cps` and `Heuristic-ops` were implemented in Java as well as extensions to the IP model for SPA to SPA-PL (Constraints 2a and 2b in Section 7.4) in CPLEX. The objective was to evaluate how well the heuristics perform (with respect to runtime and the size and degree of the matchings produced) against optimal solutions obtained from CPLEX.

Random SPA-PL instances were generated and solved using both the heuristics and the CPLEX IP implementation. We also implemented a third *naïve* heuristic (`Heuristic-naïve`). In this algorithm the project lower quotas are dropped, the resulting problem is solved and any remaining violating projects are closed. The number of students in the generated instances was varied from $n_1 = 100$ to $n_1 = 1000$ in steps of 100. For each value of n_1 considered, 1000 random instances were generated and solved using

all four techniques. For all the instances generated, $R = 5$, $n_2 = 0.3n_1$, $n_3 = 0.2n_1$ and $C_{\mathcal{P}} = C_{\mathcal{L}} = n_1$. We set the total project and lecturer lower quotas to $L_{\mathcal{P}} = 0.9n_1$ and $L_{\mathcal{L}} = 0$ respectively. The values of R, n_2, n_3, $C_{\mathcal{P}}$ and $C_{\mathcal{L}}$ are considerably different from the default ones used in the experiments above and have been chosen in order to amplify the effect of enforcing project lower quotas in the SPA-PA-1 context on the size criterion in particular.

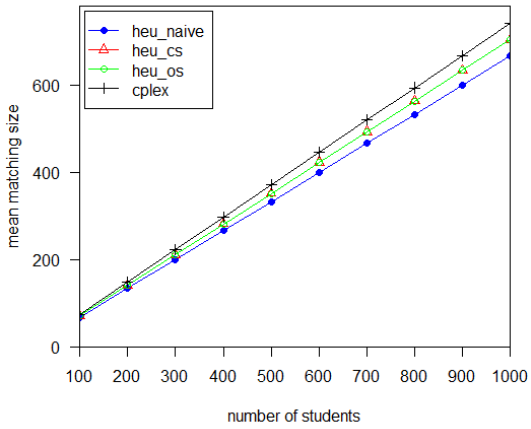

Figure 7.5: Mean matching size vs n_1

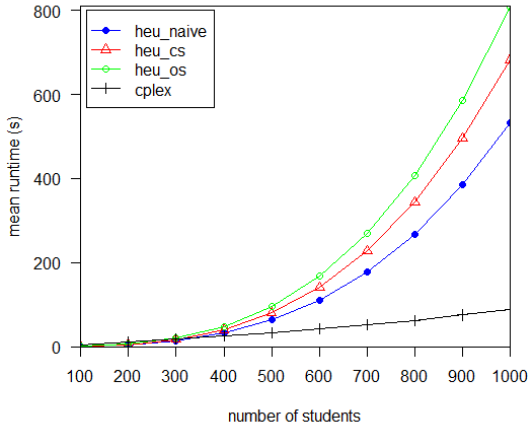

Figure 7.6: Mean runtime vs n_1

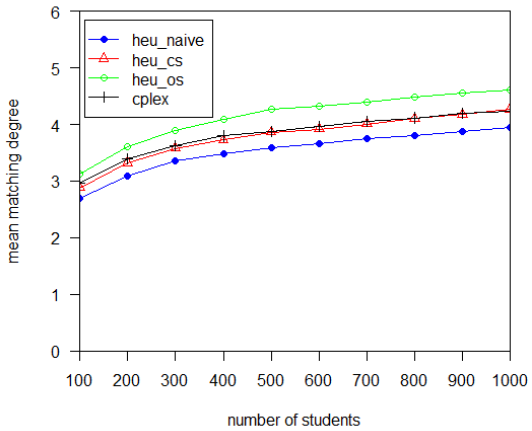

Figure 7.7: Mean matching degree vs n_1

Figure 7.5 and Table 7.2 show how the mean matching size varied with n_1. Both `Heuristic-cps` and `Heuristic-ops` found matchings that are higher than the naive approach but still noticeably lower than the optimal values. The results also show that both heuristics seem to perform equally in terms of the size of the matchings produced. Figure 7.6 shows how long it took to solve the generated instances using all four techniques. As expected the IP method was the fastest given the low value of R. Since the naive heuristic does not do much more than running the `Greedy-max-spa` algorithm, it runs the fastest of the three heuristics. The main result here is the difference in

runtime between `Heuristic-cps` and `Heuristic-ops` with `Heuristic-cps` running faster. Although the matchings produced by the heuristics need not be optimal with respect to either profile-based optimality criteria, Figure 7.7 gives us an indication of how well the heuristics perform with respect to profile. It shows a variation in the mean matching degree as n_1 was increased. We observe that `Heuristic-cps` typically produces matchings with same degree typically as CPLEX although CPLEX matchings will be larger. On the other hand, even though `Heuristic-ops` matchings are typically smaller than CPLEX, their degree is worse. Naïve matchings are typically much smaller thus explaining their lower degree.

n_1	`Heuristic-naïve`	`Heuristic-ops`	`Heuristic-cps`	CPLEX
100	67.02	70.38	70.39	74.23
200	133.33	140.55	140.57	148.32
300	200.14	211.00	211.00	222.52
400	266.37	281.18	281.20	296.58
500	332.47	351.27	351.31	370.60
600	399.63	422.14	422.20	445.07
700	466.45	492.58	492.65	519.31
800	532.49	562.64	562.81	593.24
900	599.42	633.05	633.17	667.47
1000	665.60	703.09	703.23	741.26

Table 7.2: Mean sizes of matchings produced by various heuristics and CPLEX

7.6 Conclusion

The SPA variants considered in this chapter have involved the addition of lower quotas to lecturers and projects. We have shown how these problems can be solved using our network flow models. Other extensions to SPA that may be considered interesting include:

1. allowing multiple lecturers to offer the same project (although only one of them will supervise the project in practice);
2. allowing students to be assigned to multiple projects;
3. extending the problem beyond students, lecturers and projects to consider additional sets of agents.

The following open problems relating to the variants presented in this chapter are also relevant.

1. Find faster algorithms for the minimum span problems.
2. Find an approximation algorithm for MAX SPA-PL (although we already know it cannot be approximated to within a constant factor (see Theorem 7.3.3)).

Chapter 8

Further Experimental Results for SM and SR

8.1 Introduction

As stated in Sections 2.2.2 and 2.4.2, the set of stable matchings in a given instance of SM or SR satisfies an important structural relationship with the rotation poset: that is, the stable matchings are in 1-1 correspondence with the closed (complete) subsets of the rotation poset. This structure can be exploited in order to obtain fast algorithms for a number of problems including producing the set of all stable matchings, as well as constructing other types of"fair" stable matchings. Detailed descriptions of these algorithms can be found in [38]. However little work has been done on empirically evaluating implementations of these algorithms using randomly-generated data.

In this chapter we discuss the implementation and evaluation of some of these algorithms with the aim of measuring the quantity, egalitarian cost and minimum regret of stable matchings obtained from randomly generated SM and SR instances of varying sizes. In particular we focus on the algorithms for finding the set of all stable matchings given SM and SR instances. These algorithms also enable us generate the set of stable pairs, the set of egalitarian stable matchings as well as the set of all rotations these instances admit. In Section 8.2, we describe the algorithms we implemented as well as a tool for visualising the structures they produce (i.e., rotation posets, rotation digraphs and Hasse diagrams). Section 8.3 shows results of an empirical evaluation done on the implemented algorithms and presents some interesting observations from the results obtained.

8.2 Implementation

We implemented two known algorithms (referred to here as `AlgBreakmarriage` and `GetAll StableMatchingsSM`) for finding the set of all stable matchings given an SM instance [92, 36], as well as an algorithm (referred to here as `GetAllStableMatchingsSR`) for finding the set of all stable matchings given an SR instance [38]. These implementations were done in Java using a toolkit that allows us to rapidly build matching algorithms using pre-build modules. We also extended this matching toolkit to allow for the visualisation of the structures discovered during the execution of these algorithms. These structures included the rotation poset, rotation digraph and Hasse diagram. Figure 8.1 is a screenshot of this visualisation tool. It shows the rotation poset for an SM instance displayed as an interactive graph. In this instance there are 10 rotations with $\rho \succ \sigma$ being illustrated by the edge (ρ, σ) in the digraph shown.

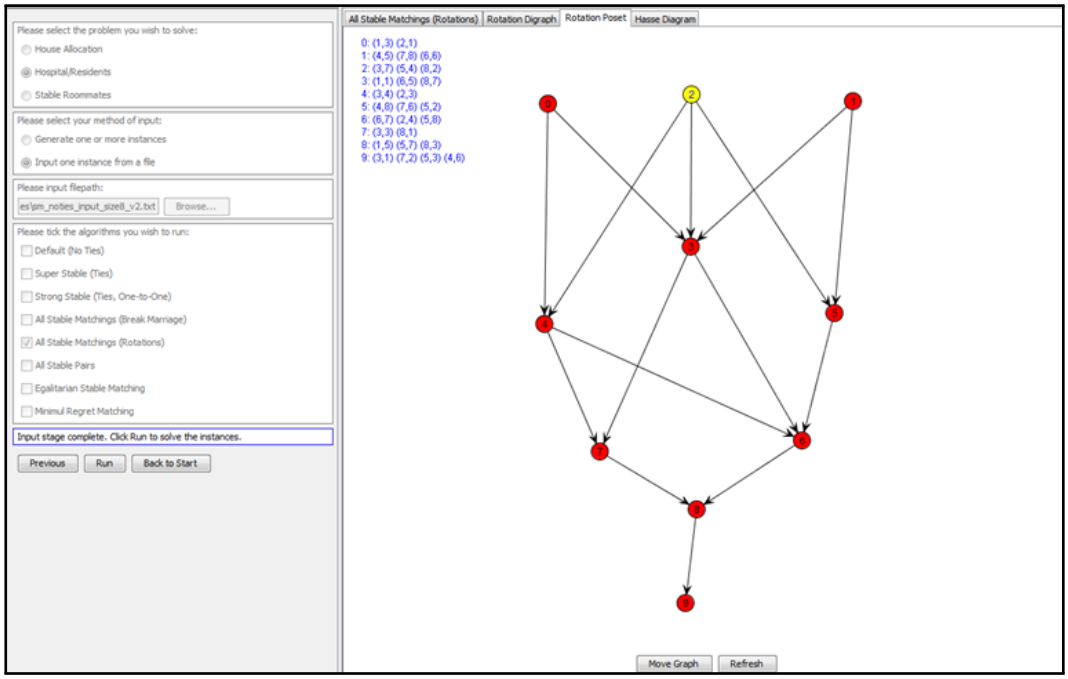

Figure 8.1: Visualising the rotation poset of an SM instance

In the following subsections we present a high-level description of the three implemented algorithms.

8.2.1 The `AlgBreakmarriage` algorithm

Let $\mathcal{S}$ be the set of all stable matchings in an SM instance and let M_0 be the man-optimal stable matching in $\mathcal{S}$. If $|\mathcal{S}| > 1$, then in each other matching $M_k \in \mathcal{S} \backslash M_0$, there exists at least one woman w_j who prefers $M_k(w_j)$ to $M_0(w_j)$ and at least one man m_i who prefers $M_0(m_i)$ to $M_k(m_i)$. The algorithm, developed by McVitie and Wilson

to find the set of all stable matchings [92], uses a recursive modification of the Gale-Shapley algorithm in conjunction with a new operation that they call `breakmarriage`. The `breakmarriage` operation receives a man m_i and a stable matching M_k (where $M(m_i) = w_j$) as input. Firstly we remove (m_i, w_j) from M_k (leaving both m_i and w_j unassigned). Then we let m_i propose to the next woman after w_j on his preference list, thus restarting the proposal sequence of the Gale-Shapley algorithm. By doing this we try to ensure that m_i gets a worse partner than w_j with w_j getting a better partner than m_i. The proposal sequence of the Gale-Shapley algorithm continues until either (i) w_j gets a proposal from a man she prefers to m_i (with m_i matched to a woman further down his preference list) in which case a new stable matching M_{k+1} has been discovered, or (ii) some man runs out of partners to propose to, in which case the `breakmarriage` operation *fails on* m_i.

The `AlgBreakmarriage` algorithm starts by finding M_0 and then calls the `breakmarriage` operation passing M_0 and some man m_1 as input. If a stable matching is found we add it to $\mathcal{S}$ otherwise nothing is added to $\mathcal{S}$. Next `breakmarriage` is called again with M_0 and the next man m_2 as input. When all the man have passed on with M_0 to the `breakmarriage` operation, we move on to the next matching in $\mathcal{S}$ starting from the first man m_1. The process continues until all men have been passed on to the `breakmarriage` operation for each stable matching found. Although the process is guaranteed to generate all stable matchings, there is a strong possibility that, due to its recursive nature, some stable matchings will be discovered multiple times. If this is the case, extra work will have to be done in order to obtain a unique set of stable matchings. To tackle this problem, two restrictions/rules can be placed on the algorithm to prevent redundant recursive calls. The rules are as follows:

Rule 1 If the `breakmarriage` operation discovers a stable matching M_k when using man m_i as input, then subsequent calls to `breakmarriage` starting from M_k can only be carried out on men m_j where $j > i$.

Rule 2 After calling the `breakmarriage` operation with man m_i as input, if any man m_j is *involved* in the proceeding proposal sequence of the Gale-Shapley algorithm, where $j < i$ the entire `breakmarriage` operation is stopped and reported as failed. A man is *involved* if, during the execution of the Gale-Shapley algorithm, he is forced to break up with his current partner and propose to another woman further down his preference list.

These two rules ensure that no redundant recursive calls to the `breakmarriage` operation are made. The algorithm will take at least $\Omega(n^3|\mathcal{S}|/\log(|\mathcal{S}|^2))$ and at most $\Omega(n^3|\mathcal{S}|)$ time where $\mathcal{S}$ is the set of all stable matchings [36].

8.2.2 The `GetAllStableMatchingsSM` algorithm

This algorithm works by first constructing the rotation poset and rotation digraph in a given SM instance and then using the rotation digraph to find the set of all stable matchings.

The steps required for the the generation of all stable matchings are as follows:

1. Run the man-oriented and woman-oriented versions of Gale-Shapley algorithm.

2. With the man-optimal and woman-optimal stable matchings generated, run the `Algorithm A` algorithm [36, Section 3.1] to generate all possible rotations in that instance.

3. Construct the rotation poset and rotation digraph from the set of rotations generated above [38, Section 3.2].

4. Use the rotation digraph to generate all stable matchings. At any point during the execution of the algorithm the rotation digraph will have one or more sources. The algorithm starts with the man-optimal stable matching and recursively exposes and eliminates rotations (thus producing new stable matchings) as they appear as sources in the rotation digraph. These sources will be candidates for elimination from the current stable matching. After each elimination from a stable matching M_k (to produce a new stable matching M_{k+1}), the eliminated rotation, say ρ, is temporarily removed from the digraph so as to expose the next set of rotations. Subsequent eliminations are then done by recursively calling the algorithm. The rotation ρ has to be put back in the digraph so as to try eliminating other rotations that are co-sources of the digraph with ρ [38, Section 3.3].

The overall time complexity of the algorithm is $O(n^2 + n|\mathcal{S}|)$ where $\mathcal{S}$ is the set of all stable matchings.

8.2.3 The `GetAllStableMatchingsSR` algorithm

In the SR context generating the set of all stable matchings (for solvable instances) is done in the following phases.

1. Phase 1 is similar to the extended Gale-Shapley algorithm but, due to the fact that only one set of agents exists, each person will both make and receive proposals. Every proposal, and the subsequent symmetric deletions that follow, is

followed by a one-way engagement between the agent proposing and the recipient of the proposal (although this one-way engagement is in general not symmetric). The resulting set of preference lists is called the *phase-1* table. If there is any agent in the phase-1 table with an empty list, then the instance does not admit a stable matching. If, in the phase-1 table, everyone has a single entry in their list, then the matching admits a unique stable matching otherwise we proceed to phase 2 [38, Section 4.2.2].

2. Phase 2 of the algorithm involves the successive exposure and elimination of rotations until either some person ends up with an empty list (in which case the instance does not admit a stable matching) or everyone has a single entry in their preference list which corresponds to a stable matching [38, Section 4.2.3].

3. During the Phase 2 all the rotations discovered are stored. The complete set of rotations can then be generated from these stored ones [38, Section 4.3].

4. Once all the rotations have been discovered and classified as singular or non-singular, finding the set of all stable matchings involves firstly eliminating all the singular rotations to obtain a *reduced phase-1* table, then the elimination the closed complete subsets of the rotation poset, each subset leading to a stable matching [38, Section 4.4].

The complexity of the algorithm is $O(n^3 \log n + n^2|\mathcal{S}|)$ where $\mathcal{S}$ represents the set of all stable matchings.

8.3 Empirical Evaluation

In this section we present results obtained from the evaluation of the `AlgBreakmarriage`, `GetAllStableMatchingsSM` and `GetAllStableMatchingsSR` algorithms using randomly-generated data. We tested the correctness of our implementations by comparing the set of stable matchings they produced with those obtained from brute-force techniques. In the case of SM we also compared results obtained from both the `AlgBreakmarriage` and `GetAllStableMatchingsSM` algorithms. Our main objective was to investigate certain properties of the stable matchings produced by these algorithms such as the number of stable matchings, rotations and stable pairs an SM or SR instance admits. For some of the algorithms, no empirical results of the nature produced exists in the literature.

In the two experiments conducted, we generated SM and SR instances (with complete and strictly-ordered preference lists) of increasing size and solved them using the

`GetAllStableMatchingsSM` and `GetAllStableMatchingsSR` algorithms respectively (the `AlgBreakmarriage` algorithm was only used for correctness testing). For each value of n in the range $n \in \{20, 40, 60, ..., 100, 200, 300, ..., 800\}$ we generated and solved $10,000$ random SM instances where n is the size of an SM instance. Also for each value of n in the range $n \in \{20, 40, 60, ..., 100, 200, 300, ..., 1000\}$ we generated and solved $10,000$ random SR instances where n is the size an SR instance. For both algorithms, we recorded, for each instance size, the average number of stable matchings, stable pairs and rotations, as well as the average cost of the egalitarian, minimum regret and maximum regret stable matchings obtained. Although more efficient algorithms exist in the literature for finding egalitarian and minimum regret stable matchings, they were not implemented here because these matchings can be obtained a by-product of generating all stable matchings. Experiments were carried out on a Windows machine with 4 Intel(R) Core(R) i5-2400 CPUs at 3.1GHz and 8GB RAM. We now present the results obtained and discuss some interesting observations made from them.

1. Figure 8.2 shows the average size of the set of stable matchings is proportional to $n \ln n$. It is known that every instance of SM admits at least one stable matching [29] and a single SM instance can admit more than one stable matching. It would thus be logical to investigate the correlation between the size of the problem instance and the number of stable matchings it is likely to admit. Lennon and Pittel showed in [81] that the expected number of stable matchings in a random SM instance of size n taken uniformly out of the $(n!)^{2n}$ possibilities is of the order $n \ln n$. Results obtained from our experiment confirm this.

   In the SR case, solvable instances can also admit multiple stable matchings. Figure 8.3 shows that the mean number of stable matchings (taken over the solvable instances) only slightly increases as we increase the instance size. This is in contrast to the SM case where larger instances tend to admit many more stable matchings. Also the relatively small numbers of stable matchings (even for instances containing 1000 agents) shed some light on how restrictive the stability criterion can be in the SR case.

2. It is also interesting to note how the average number of stable pairs varies with the size of SM and SR instances. A larger number of stable pairs is desirable as this might imply a larger set of stable matchings to choose from. With n^2 possible SM pairs and $n(n-1)/2$ possible SR pairs in an instance of size n, Figures 8.4 and 8.5 show that the average number of stable pairs is quite low. These results may be considered as further evidence of the strictness of the stability criteria (and relaxing the stability criterion has motivated results presented in Chapters

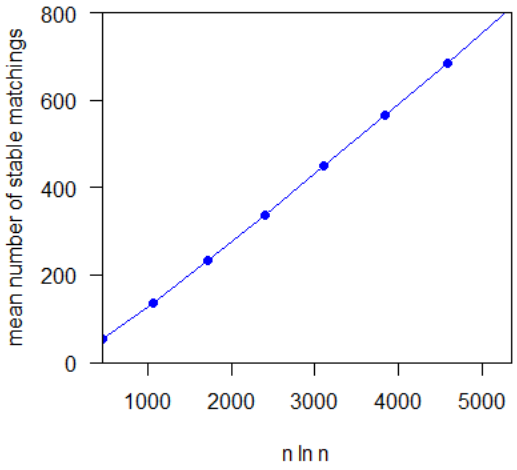

Figure 8.2: Mean $|\mathcal{S}|$ vs $n \ln n$ (SM)

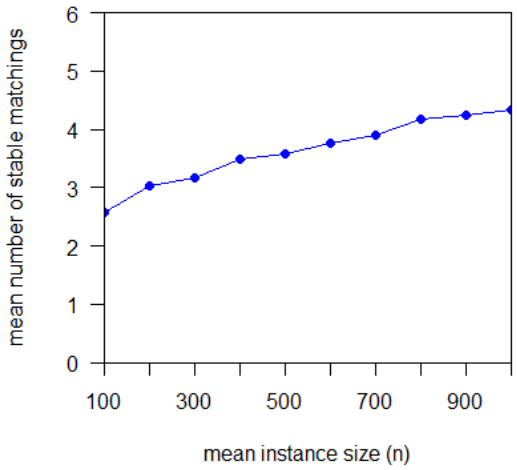

Figure 8.3: Mean $|\mathcal{S}|$ vs n (SR)

3 - 5 of this thesis). The lower the number of stable pairs, the fewer the number of stable matchings that the problem instance is likely to admit.

For SM instances, although the values increasingly deviate from n as n grows, they are still considerably smaller than the total numbers of pairs in the instances. In the SR case however, the value, which is almost exactly $n/2$, seems to agree with the observation in Figure 8.3 that random SR instances are likely to admit very few stable matchings even if the instance size is increased.

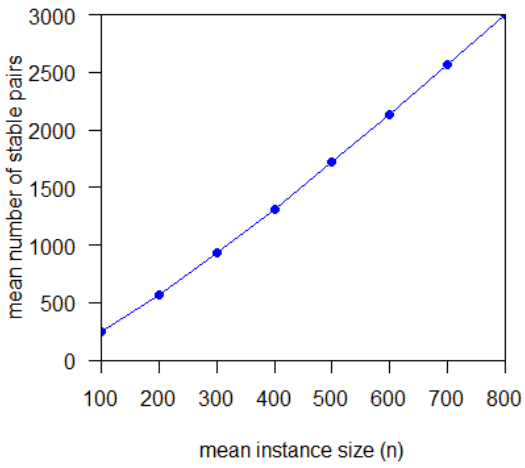

Figure 8.4: Mean stable pairs vs n (SM)

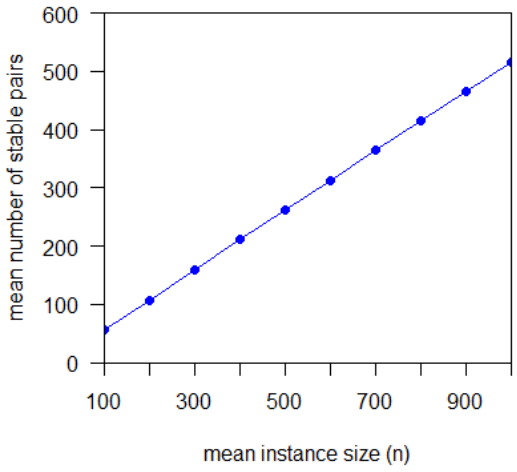

Figure 8.5: Mean stable pairs vs n (SR)

3. The rate at which the average egalitarian cost changes with the instance size is also an interesting trend to observe. The average egalitarian cost, taken over multiple SM and SR instances of a given size, gives us an indication of how fair stable matchings of that size can be. The worst case value of the cost of stable matchings in SM and SR is $\Omega(n^2)$. For the SM case, we expect the cost of an egalitarian stable matching to be better than that of both the man-optimal and

woman-optimal stable matchings. In Figure 8.6 we observe that the mean cost of the man-optimal and woman-optimal stable matchings are approximately equal and rise much more steeply than the mean cost of an egalitarian stable matching. For the SR case, Figure 8.7 shows that the average egalitarian cost grows much less steeply than $n(n-1)$ (the worst case cost of any matching) as we increase n. Both results suggest that employing the egalitarian optimality criterion seems to grow in importance as we increase the size of the SM and SR instances.

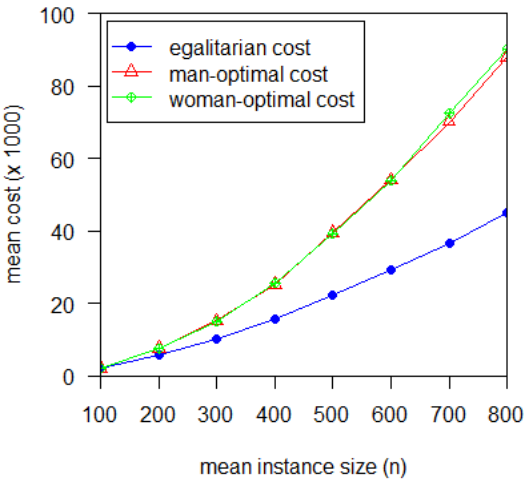

Figure 8.6: Mean cost vs n (SM)

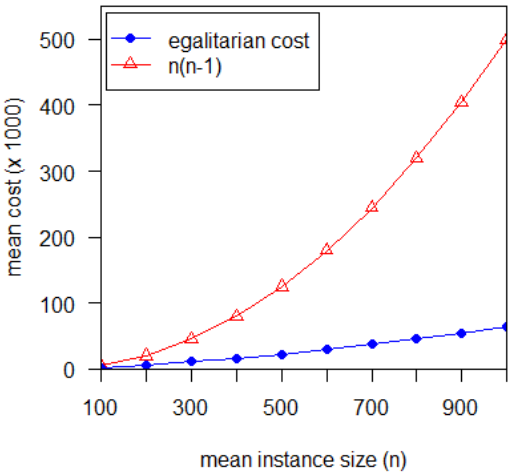

Figure 8.7: Mean cost vs n (SR)

4. Figures 8.8 and 8.9 show the mean minimum and maximum regret of the stable matchings produced as we varied the SM and SR instance sizes. For the SM case, we see the maximum regret increasing steeply with the instance size. This is not surprising as it is reasonable to expect, for example, that an SM instance of size 800 may admit some matching (in particular, the woman-optimal stable matching) with at least one man who is matched to a woman that is almost at the end of his preference list. The minimum regret curve increases much less steeply as we increase the instance size. This means that even large SM instances with 800 men and 800 women are likely to admit stable matchings that match all men and women to their top 200th choice partner. This motivates the adoption of this optimality criterion in practice.

   For the SR case we do not see a considerable gap between the mean minimum and maximum regret taken over all stable matchings. This is probably due to the fact that SR instances are likely to admit only few stable matchings. This small gap seems to suggest that, at least in practice, computing minimum regret stable matchings may be of little value. We also observe that both curves grow steeply with the instance size (although not as steeply as the mean minimum regret measure in the SM case).

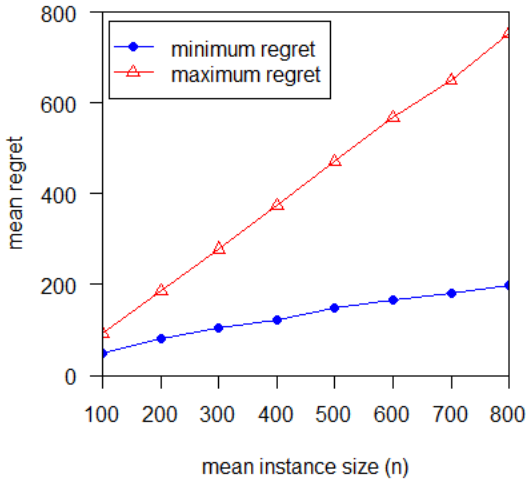

Figure 8.8: Mean regret vs n (SM)

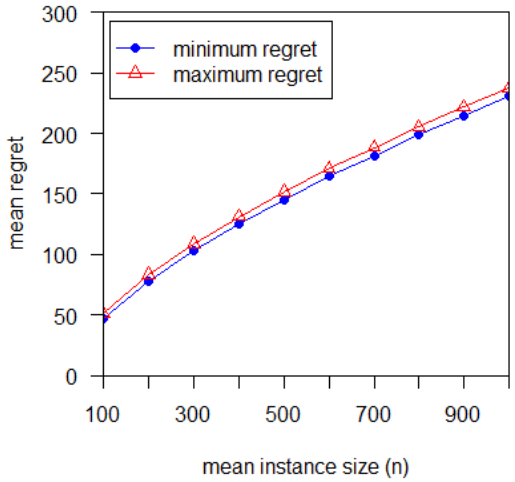

Figure 8.9: Mean regret vs n (SR)

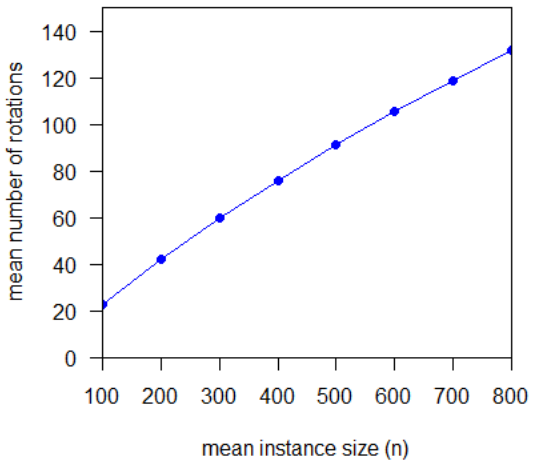

Figure 8.10: Mean number of rotations vs n (SM)

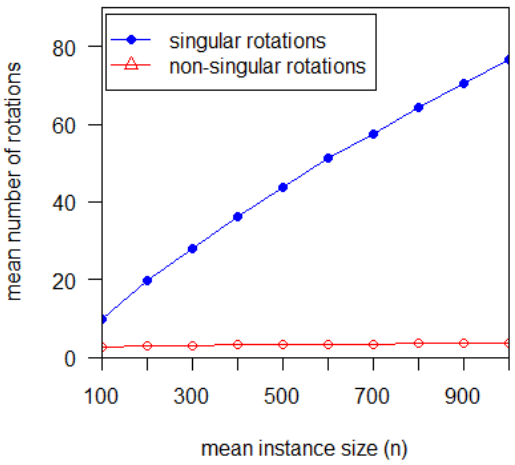

Figure 8.11: Mean number of rotations vs n (SR)

5. Investigating the way the average number of rotations varies with the size of the instances can also be important. In Figures 8.10 and 8.11, we see the number of rotations increase with the size of the SM and SR instances respectively. In the SM case, this is expected as the rise in the number of rotations corresponds to an increase in the number of stable matchings as n increases. In the SR case we observe that the steeply rising number of singular rotations has no impact on the number of stable matchings the instance admits (as singular rotations are only used to produce the reduced phase 1 table). The number of stable matchings is affected by the number of non-singular rotations the instance admits and Figure 8.11 shows this to be fairly constant as the instance size increases.

Bibliography

[1] A. Abdulkadiroğlu, P.A. Pathak, and A.E. Roth. The Boston public school match. *American Economic Review*, 95(2):368–371, 2005.

[2] A. Abdulkadiroğlu, P.A. Pathak, and A.E. Roth. The New York City high school match. *American Economic Review*, 95(2):364–367, 2005.

[3] A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[4] H.G. Abeledo and U.G. Rothblum. Stable matchings and linear inequalities. *Discrete Applied Mathematics*, 54:1–27, 1994.

[5] D. J. Abraham. Algorithmics of two-sided matching problems. Master's thesis, University of Glasgow, Department of Computing Science, 2003.

[6] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In *Proceedings of EC '07: the 8th ACM Conference on Electronic Commerce*, pages 295–304. ACM, 2007.

[7] D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn. Pareto optimality in house allocation problems. In *Proceedings of ISAAC 2004: the 15th Annual International Symposium on Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2004.

[8] D. J. Abraham, R.W. Irving, and D. F. Manlove. Two algorithms for the Student-Project allocation problem. *Journal of Discrete Algorithms*, 5(1):79–91, 2007.

[9] A. H. Abu El-Atta and M. I. Moussa. Student project allocation with preference lists over (student,project) pairs. In *Proceedings of ICCEE 09: the Second International Conference on Computer and Electrical Engineering*, pages 375–379. IEEE, 2009.

[10] A. A. Anwar and A. S. Bahaj. Student project allocation using integer programming. *IEEE Transactions on Education*, 46(3):359–367, August 2003.

[11] E. Arcaute and S. Vassilvitskii. Social networks and stable matchings in the job market. In *Proceedings of WINE '09: the 5th International Workshop on Internet and Network Economics*, volume 5929 of *Lecture Notes in Computer Science*, pages 220–231. Springer, 2009.

[12] G. Askalidis, N. Immorlica, A. Kwanashie, D. F. Manlove, and E. Pountourakis. Socially Stable matchings in the Hospitals/Residents problem. In *Proceedings of WADS '13: The 13th International Algorithms and Data Structures Symposium*, volume 8037 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2013.

[13] G. Askalidis, N. Immorlica, and E. Pountourakis. Socially stable matchings. Technical Report 1302.3309, Computing Research Repository, Cornell University Library, 2013. Available from `http://arxiv.org/abs/1302.3309`.

[14] P. Berman, M. Karpinski, and Alexander D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. Electronic Colloquium on Computational Complexity Report, number 49, 2003.

[15] P. Biró, D. F. Manlove, and S. Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.

[16] P. Biró, D. F. Manlove, and R. Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange. *Discrete Mathematics, Algorithms and Applications*, 1(4):499–517, 2009.

[17] P. Biró and I. McBride. Integer programming methods for special college admissions problems. In *Proceedings of COCOA '14: the 8th Annual International Conference on Combinatorial Optimization and Applications*, pages 429–443. Springer, 2014.

[18] E. L. Johnson C. Barnhart, M. W. P. Savelsbergh Nemhauser, L. George, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[19] K. Cechlárová and T. Fleiner. Stable roommates with free edges. Technical Report 2009-01, Egerváry Research Group on Combinatorial Optimization, Budapest, 2009.

[20] K. Cechlárová, T. Fleiner, D. F. Manlove, I. McBride, and E. Potpinková. Modelling practical placement of trainee teachers to schools. *Central European Journal of Operations Research*, pages 1–16, 2014.

[21] D. Chakrabarty and C. Swamy. Welfare maximization and truthfulness in mechanism design with ordinal preferences. In *Proceedings of ITCS '14: the 5th Conference on Innovations in Theoretical Computer Science*, pages 105–120. ACM, 2014.

[22] C.T. Cheng and E. McDermid. Maximum locally stable matchings. In *Proceedings of MATCH-UP '12: the 2nd International Workshop on Matching Under Preferences*, pages 51–62, 2012.

[23] Á. Cseh. Personal communication, 2014.

[24] R. Duan and H.-H. Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of SODA '12: the 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1413–1424. ACM-SIAM, 2012.

[25] J. Dye. A constraint logic programming approach to the stable marriage problem and its application to student-project allocation. BSc Honours project dissertation, University of York, Department of Computer Science, 2001.

[26] S. Safra E. Hazan and O. Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, May 2006.

[27] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[28] H.N. Gabow and R.E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.

[29] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[30] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[31] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA., 1979.

[32] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[33] N. Garg, T. Kavitha, A. Kumar, K. Mehlhorn, and J. Mestre. Assigning papers to referees. *Algorithmica*, 58(1):119–136, 2010.

[34] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Local search for stable marriage problems. In V. Conitzer and J. Rothe, editors, *Proceedings of COMSOC 2010: the 3rd International Workshop on Computational Social Choice*, pages 367–378. Düsseldorf University Press, 2010.

[35] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Local search for stable marriage problems with ties and incomplete lists. In *Proceedings of PRICAI 2010: the 11th Pacific Rim Conference on Artificial Intelligence*, volume 6230 of *Lecture Notes in Artificial Intelligence*, pages 64–75. Springer, 2010.

[36] D. Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

[37] D. 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.

[38] D. Gusfield and R.W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[39] M. Halldorsson, R.W. Irving, K. Iwama, D. F. Manlove, S. Miyazaki, Y. Morita, and S. Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306(1-3):431–447, September 2003.

[40] M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation of the stable marriage problem. *ACM Transactions on Algorithms*, 3(3), 2007. Article number 30.

[41] M.M. Halldórsson, R.W. Irving, K. Iwama, D. F. Manlove, S. Miyazaki, Y. Morita, and S. Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306(1-3):431–447, 2003.

[42] K. Hamada, K. Iwama, and S. Miyazaki. An improved approximation lower bound for finding almost stable stable maximum matchings. *Information Processing Letters*, 109(18):1036–1040, 2009.

[43] P.R. Harper, V. de Senna, I.T. Vieira, and A.K. Shahani. A genetic algorithm for the project assignment problem. *Computers and Operations Research*, 32:1255–1265, 2005.

[44] M. Hoefer. Local matching dynamics in social networks. *Information and Computation*, 222:20–35, 2013.

[45] M. Hoefer and L. Wagner. Locally stable marriage with strict preferences. In *Proceedings of ICALP 2013: the 40th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science. Springer, 2013.

[46] J.E. Hopcroft and R.M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[47] C.-C. Huang and T. Kavitha. Efficient algorithms for maximum weight matchings in general graphs with small edge weights. In *Proceedings of SODA '12: the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1400–1412. SIAM, 2012.

[48] C.-C. Huang, T. Kavitha, K. Mehlhorn, and D. Michail. Fair matchings and related problems. In *Proceedings of FSTTCS 2013: the 33rd International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 24, pages 339–350. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[49] A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979.

[50] R.W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6:577–595, 1985.

[51] R.W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.

[52] R.W. Irving. Matching medical students to pairs of hospitals: a new variation on a well-known theme. In *Proceedings of ESA '98: the Sixth European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 381–392. Springer, 1998.

[53] R.W. Irving. Greedy matchings. Technical Report TR-2003-136, University of Glasgow, Department of Computing Science, 2003.

[54] R.W. Irving. Greedy and generous matchings via a variant of the Bellman-Ford algorithm. Unpublished manuscript, 2006.

[55] R.W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.

[56] R.W. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[57] R.W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34(3):532–543, 1987.

[58] R.W. Irving and D. F. Manlove. The Stable Roommates Problem with Ties. *Journal of Algorithms*, 43:85–105, 2002.

[59] R.W. Irving and D. F. Manlove. Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems. *Journal of Combinatorial Optimization*, 16(3):279–292, 2008.

[60] R.W. Irving and D. F. Manlove. Finding large stable matchings. *ACM Journal of Experimental Algorithmics*, 14, 2009. Section 1, article 2, 30 pages.

[61] R.W. Irving, D. F. Manlove, and G. O'Malley. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms*, 7(2):213–219, 2009.

[62] R.W. Irving, D. F. Manlove, and S. Scott. The Hospitals/Residents problem with Ties. In *Proceedings of SWAT 2000: the 7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2000.

[63] R.W. Irving, D. F. Manlove, and S. Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008.

[64] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. In *Proceedings of ICALP '99: the 26th International Colloquium on Automata, Languages, and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 1999.

[65] K. Iwama, S. Miyazaki, and K. Okamoto. A $\left(2 - c\frac{\log n}{n}\right)$-approximation algorithm for the stable marriage problem. In *Proceedings of SWAT 2004: the 9th Scandinavian Workshop on Algorithm Theory*, volume 3111 of *Lecture Notes in Computer Science*, pages 349–361. Springer, 2004.

[66] K. Iwama, S. Miyazaki, and K. Okamoto. A $\left(2 - c\frac{\log n}{n}\right)$-approximation algorithm for the stable marriage problem. *IEICE Transactions on Information and Systems*, E89-D(8):2380–2387, 2006. Preliminary version appeared in [65].

[67] K. Iwama, S. Miyazaki, and N. Yamauchi. A 1.875–approximation algorithm for the stable marriage problem. In *Proceedings of SODA '07: the Eighteenth ACM/SIAM Symposium on Discrete Algorithms*, pages 288–297. ACM-SIAM, 2007.

[68] K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. *Journal of Discrete Algorithms*, 13:59–66, 2012.

[69] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[70] B. A. Kassa. A linear programming approach for placement of applicants to academic programs. *SpringerPlus*, 2(1):1–7, 2013.

[71] A. Kato. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics*, 10:1–19, 1993.

[72] D. Kazakov. Co-ordination of student-project allocation. Manuscript, University of York, Department of Computer Science, 2002.

[73] Z. Király. Better and simpler approximation algorithms for the stable marriage problem. In *Proceedings of ESA '08: the 16th Annual European Symposium on Algorithms*, volume 5193 of *Lecture Notes in Computer Science*, pages 623–634. Springer, 2008.

[74] Z. Király. Better and simpler approximation algorithms for the stable marriage problem. *Algorithmica*, 60:3–20, 2011. Preliminary version appeared in [73].

[75] Z. Király. Linear time local approximation algorithm for maximum stable marriage. In *Proceedings of MATCH-UP '12: the 2nd International Workshop on Matching Under Preferences*, pages 99–110, 2012.

[76] D.E. Knuth. *Mariages Stables*. Les Presses de L'Université de Montréal, 1976. English translation in *Stable Marriage and its Relation to Other Combinatorial Problems*, volume 10 of CRM Proceedings and Lecture Notes, American Mathematical Society, 1997.

[77] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. In *Annals of Discrete Mathematics*, volume 2, pages 65–74. North-Holland, 1978.

[78] E. Kujansuu, T. Lindberg, and E. Mäkinen. The stable roommates problem and chess tournament pairings. *Divulgaciones Matemáticas*, 7(1):19–28, 1999.

[79] A. Kwanashie and D. F. Manlove. An Integer Programming approach to the Hospitals/Residents problem with Ties. Technical Report 1308.4064, Computing Research Repository, Cornell University Library, 2013.

[80] G. Han L. Pan, S. C. Chu and J. Z. Huang. Multi-criteria student project allocation: A case study of goal programming formulation with dss implementation. In *Proceedings of ISORA 2009: The Eighth International Symposium on Operations Research and Its Applications, Zhangjiajie, China*, pages 75–82, 2009.

[81] C. Lennon and B. Pittel. On the likely number of solutions for the stable marriage problem. *Combinatorics Probability and Computing*, 18(3):371–421, May 2009.

[82] K. S. M. Sahari M. H. Hasan and A. Anuar. Implementation of a new preference based final year project title selection system for undergraduate engineering students in uniten. In *in Proceedings of ICEED 2009: International Conference on Engineering Education*, pages 230–235, Dec 2009.

[83] D. Maier and J.A. Storer. A note on the complexity of the superstring problem. In *12th Annual Conference on Information Sciences and Systems*, pages 52–56, 1978.

[84] D. F. Manlove. Stable marriage with ties and unacceptable partners. Technical Report TR-1999-29, University of Glasgow, Department of Computing Science, January 1999.

[85] D. F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[86] D. F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[87] D. F. Manlove and G. O'Malley. Student project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6:553–560, 2008.

[88] D. Marx and I. Schlotter. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica*, 58(1):170–187, 2010.

[89] I. McBride and D. F. Manlove. An integer programming model for the Hospitals/Residents problem with Couples. In *Proceedings of OR 2013: the International Conference on Operations Research*, pages 293–299. Springer, 2014.

[90] E. McDermid. A 3/2 approximation algorithm for general stable marriage. In *Proceedings of ICALP '09: the 36th International Colloquium on Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 689–700. Springer, 2009.

[91] E. McDermid and R.W. Irving. Sex equal stable matchings: Complexity and exact algorithms. *Algorithmica*, 2013.

[92] D.G. McVitie and L.B. Wilson. The stable marriage problem. *Communications of the ACM*, 14(7):486–490, 1971.

[93] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. Unpublished manuscript, 2006.

[94] S. Micali and V.V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of FOCS '80: the 21st Annual IEEE Symposium on Foundations of Computer Science*, pages 17–27. IEEE Computer Society, 1980.

[95] J. B. Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings STOC '13: the 45th Annual ACM Symposium on the Theory of Computing*, pages 765–774. ACM, 2013.

[96] J.B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[97] K.E. Paluch. Faster and simpler approximation of stable matchings. In *Proceedings of WAOA '11: 9th Workshop on Approximation and Online Algorithms*, Lecture Notes in Computer Science. Springer, 2012.

[98] N. Perach, J. Polak, and U.G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion. *International Journal of Game Theory*, 36(3-4):519–535, 2008.

[99] A. Podhradský. Stable marriage problem algorithms. Master's thesis, Master's thesis, Masaryk University, Faculty of Informatics, 2010. Available from `http://is.muni.cz/th/172646/fi_m`.

[100] A. Podhradsky. Approximative algorithms for the problem of stable pairing. 2011 [cit. 26/10/2012].

[101] A.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.

[102] A.E. Roth, U.G. Rothblum, and J.H. Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of Operations Research*, 18(4):803–828, 1993.

[103] A.E. Roth, T. Sönmez, and M.U. Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, 2004.

[104] A.E. Roth, T. Sönmez, and M.U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005.

[105] A.E. Roth, T. Sönmez, and M.U. Ünver. Efficient kidney exchange: Coincidence of wants in a market with compatibility-based preferences. *American Economic Review*, 2007.

[106] A.E. Roth and M.A.O. Sotomayor. *Two-sided matching: a study in game-theoretic modeling and analysis*, volume 18 of *Econometric Society Monographs*. Cambridge University Press, 1990.

[107] A.E. Roth and X. Xing. Jumping the gun: imperfections and institutions related to the timing of market transactions. *American Economic Review*, 84(4):992–1044, 1994.

[108] U.G. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.

[109] H. M. Saber and J. B. Ghosh. Assigning students to academic majors. *Omega*, 29(6):513 – 523, 2001.

[110] C.T.S. Sng. *Efficient Algorithms for Bipartite Matching Problems with Preferences*. PhD thesis, University of Glasgow, Department of Computing Science, 2008.

[111] C. Y. Teo and D. J. 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.

[112] M. Thorn. A constraint programming approach to the student-project allocation problem. BSc Honours project dissertation, University of York, Department of Computer Science, 2003.

[113] S. Varone and D. Schindl. Course opening, assignment and timetabling with student preferences. In *Proceedings of ICORES: International Conference on Operations Research and Enterprise Systems*, 2013.

[114] J.H. Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

[115] H. Yanagisawa. *Approximation Algorithms for Stable Marriage Problems*. PhD thesis, Kyoto University, School of Informatics, 2007.

[116] M. Zelvyte. The student-project allocation problem using network flow. BSc Honours project dissertation, University of Glasgow, School of Mathematics and Statistics, 2014.

[117] L. Zhou. On a conjecture by Gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990.