# Efficient Algorithms for Bipartite Matching Problems with Preferences

by

**Colin Thiam Soon Sng**

A thesis submitted to the
Faculty of Information and Mathematical Sciences
at the University of Glasgow
for the degree of
Doctor of Philosophy

June 2008

## Abstract

Matching problems involve a set of participants, where each participant has a capacity and a subset of the participants rank a subset of the others in order of preference (strictly or with ties). Matching problems are motivated in practice by large-scale applications, such as automated matching schemes, which assign participants together based on their preferences over one another.

This thesis focuses on *bipartite matching problems* in which there are two disjoint sets of participants (such as medical students and hospitals). We present a range of efficient algorithms for finding various types of optimal matchings in the context of these problems. Our optimality criteria involve a diverse range of concepts that are alternatives to classical stability. Examples include so-called *popular* and *Pareto optimal* matchings, and also matchings that are optimal with respect to their *profile* (the number of participants obtaining their first choice, second choice and so on).

The first optimality criterion that we study is the notion of a *Pareto optimal matching*, a criterion that economists regard as a fundamental property to be satisfied by an optimal matching. We present the first algorithmic results on Pareto optimality for the Capacitated House Allocation problem (CHA), which is a many-to-one variant of the classical House Allocation problem, as well as for the Hospitals-Residents problem (HR), a generalisation of the classical Stable Marriage problem. For each of these problems, we obtain a characterisation of Pareto optimal matchings, and then use this to obtain a polynomial-time algorithm for finding a maximum Pareto optimal matching.

The next optimality criterion that we study is the notion of a *popular matching*. We study popular matchings in CHA and present a polynomial-time algorithm for finding a maximum popular matching or reporting that none exists, given any instance of CHA. We extend our findings to the case in CHA where preferences may contain ties (CHAT) by proving the extension of a well-known result in matching theory to the capacitated bipartite graph case, and using this to obtain a polynomial-time algorithm for finding a maximum popular matching, or reporting that none exists.

We next study popular matchings in the Weighted Capacitated House Allocation problem (WCHA), which is a variant of CHA where the agents have weights assigned to them. We identify a structure in the underlying graph of the problem that singles out those edges that cannot belong to a popular matching. We then use this to construct a polynomial-time algorithm for finding a maximum popular matching or reporting that none exists, for

the case where preferences are strict.

We then study popular matchings in a variant of the classical Stable Marriage problem with Ties and Incomplete preference lists (SMTI), where preference lists are symmetric. Here, we provide the first characterisation results on popular matchings in the bipartite setting where preferences are two-sided, which can either lead to a polynomial-time algorithm for solving the problem or help establish that it is NP-complete. We also provide the first algorithm for testing if a matching is popular in such a setting.

The remaining optimality criteria that we study involve *profile-based optimal matchings*. We define three versions of what it means for a matching to be optimal based on its profile, namely so-called *greedy maximum*, *rank-maximal* and *generous maximum* matchings. We study each of these in the context of CHAT and the Hospitals-Residents problem with Ties (HRT). For each problem model, we give polynomial-time algorithms for finding a greedy maximum, a rank-maximal and a generous maximum matching.

# Contents


**1 A selective review of the literature** **1**

1.1 Motivation . . . . . 1

1.2 Theory of matching in bipartite graphs . . . . . 4

1.2.1 Unweighted Graphs . . . . . 4

1.2.1.1 Capacitated graphs . . . . . 5

1.2.2 Weighted Graphs . . . . . 5

1.2.3 Edmonds-Gallai Decomposition . . . . . 6

1.3 Bipartite matching problems, one-sided preference lists . . . . . 7

1.3.1 House Allocation Problem . . . . . 7

1.3.1.1 Variants of HA . . . . . 8

1.3.2 Optimality criteria for bipartite matching problems . . . . . 9

1.3.2.1 Pareto optimal matchings . . . . . 9

1.3.2.2 Popular matchings . . . . . 9

1.3.2.3 Profile-based optimal matchings . . . . . 11

1.4 Bipartite matching problems, two-sided preference lists . . . . . 12

1.4.1 One-one mapping: the classical Stable Marriage Problem . . . . . 12

1.4.1.1 Incomplete lists . . . . . 13

1.4.1.2 Ties . . . . . 14

1.4.1.3 Ties and Incomplete lists . . . . . 14

1.4.2 One-many mapping: the Hospitals-Residents Problem . . . . . 15

1.5 Non-bipartite matching problems . . . . . 17

1.5.1 Stable Roommates Problem . . . . . 17

**2 Pareto optimal matchings in CHA** **19**

2.1 Introduction . . . . . 19

2.2 Basic terminology and preliminary results . . . . . 19
2.3 Characterisation of Pareto optimal matchings . . . . . 20
2.4 Maximum Pareto optimal matchings . . . . . 24
2.4.1 Phase 2 of the algorithm . . . . . 24
2.4.2 Phase 3 of the algorithm . . . . . 26
2.4.2.1 Pre-processing step . . . . . 27
2.4.2.2 Phase 3 loop . . . . . 29
2.4.2.3 Correctness of Phase 3 loop . . . . . 30
2.4.3 Implementation and analysis . . . . . 32
2.5 Open problem . . . . . 34

**3 Pareto optimal matchings in HR** **36**
3.1 Introduction . . . . . 36
3.2 Basic terminology and preliminary results . . . . . 36
3.3 Characterisation of Pareto optimal matchings in HR . . . . . 37
3.4 Maximum Pareto optimal matchings in HR . . . . . 41
3.5 Maximum Pareto optimal matchings in SMI . . . . . 42
3.6 Open problems . . . . . 43

**4 Popular matchings in CHAT** **44**
4.1 Introduction . . . . . 44
4.2 Popular matchings in CHA . . . . . 45
4.2.1 Characterising popular matchings . . . . . 45
4.2.2 Finding a popular matching . . . . . 48
4.2.3 Finding a maximum popular matching . . . . . 49
4.2.4 “Cloning” verus our direct approach . . . . . 50
4.3 Popular matchings in CHAT . . . . . 50
4.3.1 Characterising popular matchings . . . . . 50
4.3.2 Finding a popular matching . . . . . 57
4.3.3 Finding a maximum popular matching . . . . . 60
4.3.4 “Cloning versus our direct approach . . . . . 61
4.4 Open problem . . . . . 61

**5 Popular matchings in WCHA** **62**

5.1 Introduction . . . 62

5.2 Characterising a popular matching . . . 63

5.3 Algorithm for finding a popular matching . . . 71

5.3.1 Potential improvement paths . . . 72

5.3.2 Pruning the graph . . . 73

5.3.3 Proof of correctness . . . 76

5.3.4 Finding a popular matching . . . 83

5.3.5 Finding a maximum popular matching . . . 84

5.3.6 "Cloning" versus our direct approach . . . 85

5.4 Open problem . . . 85

**6 Popular matchings in SMTI-SYM** **86**

6.1 Chapter overview . . . 86

6.2 Basic terminology and preliminary results . . . 87

6.3 Characterising popular matchings . . . 90

6.4 Structure of popular matchings . . . 98

6.4.1 Mutually exclusive edge pairs . . . 99

6.4.2 Testing a matching for popularity . . . 102

6.4.3 Concluding remarks . . . 102

**7 Profile-based optimal matchings in CHAT** **105**

7.1 Introduction . . . 105

7.2 Basic terminology . . . 106

7.3 Greedy maximum matchings . . . 107

7.3.1 Finding a greedy maximum matching . . . 107

7.3.2 Proof of correctness . . . 112

7.3.3 Time complexity analysis . . . 115

7.4 Rank-maximal matchings . . . 115

7.4.1 Finding a rank-maximal matching . . . 116

7.4.2 Proof of correctness . . . 117

7.4.3 Time complexity analysis . . . 118

7.4.4 Alternative approaches to finding a rank-maximal matching . . . 119

7.4.4.1 Reduction to the Assignment problem . . . 119

7.4.4.2 Adapting Algorithm Greedy-Max . . . 121
7.4.4.3 "Cloning" . . . 123
7.5 Generous Maximum Matchings . . . 123
7.5.1 Finding a generous maximum matching . . . 124
7.5.2 Proof of correctness . . . 125
7.5.3 Time complexity analysis . . . 126
7.6 Open Problems . . . 126

**8 Profile-based optimal matchings in HRT** **128**
8.1 Introduction . . . 128
8.2 Basic terminology . . . 129
8.3 Greedy Maximum matchings . . . 129
8.3.1 Finding a greedy maximum matching . . . 129
8.3.2 Proof of correctness . . . 131
8.3.3 Time complexity analysis . . . 132
8.4 Rank-maximal matchings . . . 133
8.4.1 Finding a rank-maximal matching . . . 133
8.5 Generous Maximum Matchings . . . 135
8.5.1 Finding a generous maximum matching . . . 135
8.6 Profile-based optimal matchings in SMTI . . . 136
8.7 Open Problems . . . 136

**9 Conclusion** **138**
9.1 Summary of thesis contribution . . . 138
9.2 Pareto optimal matchings . . . 140
9.3 Popular matchings . . . 140
9.4 Profile-based optimal matchings . . . 141
9.5 General observations . . . 141
9.6 Future work . . . 142

# List of Figures


2.1 An instance $I_1$ of CHA . . . . . 23

3.1 An instance $I_1$ of HR . . . . . 40

5.1 An instance $I_1$ of WCHA . . . . . 65

6.1 An instance $I_1$ of SMTI-SYM . . . . . 88
6.2 An instance $I_2$ of SMTI-SYM . . . . . 89
6.3 An instance $I_3$ of SMTI-SYM . . . . . 89
6.4 An instance $I_4$ of SMTI-SYM . . . . . 90
6.5 An instance $I_5$ of SMTI-SYM. . . . . 94
6.6 An instance $I_6$ of SMTI-SYM. . . . . 103

7.1 An instance $I_1$ of CHAT . . . . . 108
7.2 An instance $I_2$ of CHAT . . . . . 116

9.1 Complexity of algorithms for producing optimal matchings in bipartite matching problems with preferences. . . . . 139

# List of Algorithms


1 Classical augmenting path algorithm . . . . . 5
2 Algorithm Greedy-PaCHA . . . . . 22
3 Phase 2 loop . . . . . 25
4 Process (Q) . . . . . 27
5 Main Phase 3 loop . . . . . 29
6 Algorithm Popular-CHA . . . . . 48
7 Algorithm Popular-CHAT . . . . . 58
8 Algorithm Label-f . . . . . 64
9 First stage of Algorithm Prune-WCHA . . . . . 74
10 Second stage of Algorithm Prune-WCHA . . . . . 75
11 Algorithm Label-s . . . . . 93
12 Algorithm Greedy-Max . . . . . 107
13 Algorithm Max-Aug . . . . . 111
14 Algorithm Rank-Max . . . . . 117
15 Algorithm Greedy-Rank-Max . . . . . 122

## Declaration

This thesis is submitted in accordance with the rules for the degree of Doctor of Philosophy at the University of Glasgow in the Faculty of Information and Mathematical Sciences. None of the material contained herein has been submitted for any other degree, and all the results are claimed as original.

## Publications and papers submitted for publication

1. D.F. Manlove and C.T.S. Sng. Popular matchings in the Capacitated House Allocation Problem. In *Proceedings of ESA 2006: the 14th Annual European Symposium on Algorithms*, volume 4168 of *Lecture Notes in Computer Science*, pages 492-503, Springer-Verlag, 2006. (This paper is based on Chapter 4.)

2. C.T.S. Sng and D.F. Manlove. Popular Matchings in the Weighted Capacitated House Allocation Problem. In *Proceedings* of *ACiD 2007: the 3rd Algorithms and Complexity in Durham workshop*, volume 9 of Texts in Algorithmics, pages 129-140, *College Publications*, 2007. (This paper is based on Chapter 5. Also submitted to the *Journal of Discrete Algorithms*.)

## Acknowledgements

I would like to thank David Manlove for giving me the opportunity to undertake a PhD in algorithms. I have benefited immensely from the help and encouragement that he has given me over the last four years, in particular the meticulous approach that he has always taken towards reading my work, and I would like to express here my heartfelt gratitude to him. I would also like to thank Rob Irving, my second supervisor, for the invaluable help and advice that he has given me at several important times of my research.

My PhD studies were supported by the Department of Computing Science, University of Glasgow, and Universities UK (Overseas Research Student Award).

I would also like to thank my family for their love and support. First of all, I wish to thank my mother for enabling me to become what I am today. My wife Lillian has stood by me endlessly the last seven years, and it is with her constant stream of patience, love and encouragement that I am able to cross the finishing line. Little Grace has been a bundle of joy to me since her birth.

Last, but most importantly, I am grateful to Abba for what He has given me.

# Chapter 1

# A selective review of the literature

## 1.1 Motivation

Matching problems are motivated in practice by large-scale applications, such as automated matching schemes, which assign participants together based on their preferences over one another. In Scotland [27] and the USA [49] for example, centralised automated matching schemes, such as the Scottish Foundation Allocation Scheme (SFAS) and the National Resident Matching Program (NRMP) respectively, annually construct allocations of graduating medical students to hospital posts. In Singapore, a centralised matching mechanism is used to assign primary school students to secondary schools [61]. In Romania [36], the Netherlands [11] and the USA [52–55], systematic programs have been established for managing kidney exchange. Additionally, there are many other examples of centralised matching schemes in various countries, in educational, vocational and medical contexts.

Matching problems involve a set of participants, where each participant has a *capacity* and a subset of the participants rank a subset of the others in order of preference (strictly or with ties). The term *matching* implies the attempt to assign each participant to one or more acceptable partner(s) in some way to meet some specified criterion without exceeding the capacities of the participants. Given the large number of participants typically involved in the types of matching schemes discussed above, constructing matchings manually is time-consuming, error-prone and infeasible for large instances. Algorithms automate the process and again, given the typical sizes of input datasets, it is vital to ensure that algorithms for matching problems are as efficient as possible. In its broadest sense, the notion of efficiency involves all the various computing resources needed for executing an algorithm. The measure of efficiency that will be the prime focus of this research is the time

requirement of a given algorithm because it is often the dominant factor that determines whether or not a particular algorithm is useful in practice, regardless of potential increases in processing power [21].

Furthermore, given the applications of matching problems, and the implications of a participant's allocation in a matching for their quality of life, it is of paramount importance that the matching algorithms that drive such applications should optimise in some sense, and insofar as is possible, the satisfaction of the participants according to their preferences.

There are many ways to classify matching problems and a convenient distinction can be made between a bipartite matching model in which there are two disjoint sets of participants, and a non-bipartite model in which there is only a single set of participants. Three-dimensional matching problems (in which there are three disjoint sets of participants) have also been considered but a number of variants have been shown to be NP-complete [21, 45, 58]; thus it is unlikely that there exist efficient algorithms for the solution of such problems. In addition, matching problems may be further sub-divided according to the types of preference lists that are involved (two-sided or one-sided) as well as the kind of mapping that is being sought in order to assign the members of one side to the other, so that it is possible to classify these problems as follows:

1. Bipartite matching problems
    (a) One-sided preference lists
        i. One-one mapping, e.g., House Allocation problem
        ii. Many-one mapping, e.g., Capacitated House Allocation problem
    (b) Two-sided preference lists
        i. One-one mapping, e.g., Stable Marriage problem
        ii. Many-one mapping, e.g., Hospitals-Residents problem
2. Non-bipartite matching problems
    (a) One-one mapping, e.g., Stable Roommates problem

For bipartite matching problems with preferences, an extensively studied problem is the classical Stable Marriage Problem (SM) [17], in which the participants consist of two disjoint sets of agents, say $n$ men and $n$ women, each of whom ranks all members of the opposite sex in order of preference and a matching is just a one-one mapping

between the two sets. Note that we henceforth use the term *agents* to refer to those participants in matching problems who have preference lists. Hence, the agents in an SM instance are the men and women. Alternatively, bipartite matchings can also involve many-one mappings. For example, in the context of the Hospitals-Residents problem (HR) [17, 22], the participants are residents (graduating medical students) and hospitals, with each member of the latter set having some fixed number of "posts"(its capacity). All participants are agents since each resident ranks a subset of hospitals in order of preference and vice versa. A matching is an assignment of residents to hospitals so that no hospital exceeds its capacity. These are examples of bipartite matching problems where the preference lists are two-sided.

Alternatively, preference lists for bipartite matching problems can be one-sided. An example of this type of problem is the House Allocation problem (HA) [1, 3], where an attempt is made to allocate a set $H$ of objects (e.g., houses, posts etc) using a one-one mapping among a set $A$ of agents, each of whom ranks a subset of $H$ in order of preference. The Capacitated House Allocation problem (CHA) is a generalisation of HA in which a many-one mapping of $A$ to $H$ is sought instead. In addition to bipartite matching problems, non-bipartite matching problems are also widely studied. In the classical Stable Roommates problem (SR) [17, 25], the participants consist of a single set of agents each of whom ranks the others in order of preference, and a matching is a partition of the set into disjoint pairs of roommates.

The focus of this research will be the bipartite matching model which underpins most of the aforementioned matching schemes. We explore a diverse range of optimality concepts that are applicable to many new and also well-studied bipartite matching problems, and find efficient algorithms for constructing matchings that are optimal according to these criteria. The remainder of this chapter is structured as follows. In Section 1.2, we give a brief overview of several important results from matching theory in bipartite graphs; some of these will subsequently be used by the algorithms that we will describe for the bipartite matching problems considered in this thesis. Reviews of previous results on bipartite matching problems with one-sided preferences, bipartite matching problems with two-sided preferences and non-bipartite problems are then contained in Sections 1.3, 1.4 and 1.5 respectively.

## 1.2 Theory of matching in bipartite graphs

### 1.2.1 Unweighted Graphs

Let $G = (U, W, E)$ be a bipartite graph with $n_1$ vertices in $U$, $n_2$ vertices in $W$ and $m$ edges. Also, let $n = n_1 + n_2$. A *matching* $M$ of $G$ is a subset of $E$ such that no two edges in $M$ share a common vertex. We say that an edge $e \in E$ is *matched* if $e \in M$; otherwise, $e$ is *unmatched*. Similarly, we say that a vertex $v \in U \cup W$ is *matched* in $M$ if it is incident to an edge in $M$ or *unmatched* otherwise. We define the *cardinality* of the matching $M$, denoted by $|M|$, to be the number of edges in $M$. A matching $M$ is *maximal* if $M$ is not a proper subset of any other matching in $G$. A matching $M$ is *maximum* if $M$ contains the largest possible number of edges. Note that every maximum matching must be maximal, but the converse need not be true. A matching $M$ is *perfect* if every vertex in $U \cup W$ is matched in $M$. Given an arbitrary matching $M$, an *alternating path* is a path $P$ in which the edges of $P$ are alternatively in $M$, and not in $M$. An *augmenting path* with respect to $M$ is an alternating path whose end vertices are unmatched. The following theorem due to Berge gives one of the most fundamental results underpinning matching theory.

**Theorem 1.2.1** (**Berge** [8])**.** *Let $M$ be an arbitrary matching in $G$. Then, $M$ has maximum cardinality if and only if there is no augmenting path with respect to $M$.*

This theorem gives rise to the classical augmenting path algorithm for finding a maximum matching in any bipartite graph $G$ [46], as shown in Algorithm 1. The algorithm runs in stages where a search for an augmenting path is conducted in each stage.

Starting from the unmatched vertices in $U$, it is straightforward to see that the search for an augmenting path relative to $M$ can be organised as a restricted breadth-first search in which only edges not matched in $M$ are followed from vertices in $U$ and only edges matched in $M$ are followed from vertices in $W$, to ensure alternation. If any augmenting path exists, then it is clear to see that this search will find one, which we denote by $P$.

The algorithm then *augments* the current matching $M$ with $P$ by inverting the matched edges in $P$, i.e. the matched edges in $P$ become unmatched, and vice versa, so that we increase the cardinality of $M$ by 1. If an augmenting path does not exist, then $M$ is maximum by Theorem 1.2.1. It is easy to see that a search for an augmenting path using the method described above takes $O(n + m)$ time. Since there are at most $O(n)$ such searches, it follows that the classical augmenting path algorithm can thus be made to run in $O(n(n + m))$ time.

**Algorithm 1** Classical augmenting path algorithm

1: $M := \emptyset$;
2: **while** $G$ admits an augmenting path $P$ with respect to $M$ **do**
3: $\quad M := M \oplus P$;
4: **return** $M$;

However, faster algorithms for finding a maximum matching in a given bipartite graph exist, and the best known algorithm is due to Hopcroft and Karp [23]. Their approach is similar to the classical augmenting path algorithm but in each stage, a maximal set of vertex disjoint augmenting paths of shortest length is found and used to augment the matching instead of a single augmenting path. The importance of this is that the number of searches is reduced to at most $O(\sqrt{n})$, and the Hopcroft-Karp algorithm thus runs in $O(\sqrt{n}m)$ time. Hence, we have the following result.

**Theorem 1.2.2** (**Hopcroft and Karp** [23])**.** *Let $G = (U, W, E)$ be a bipartite graph, with $n$ vertices in $U \cup W$ and $m$ edges in $E$. Then, a maximum matching in $G$ can be found in $O(\sqrt{n}m)$ time.*

#### 1.2.1.1 Capacitated graphs

Let $G = (U, W, E)$ be a bipartite graph in which each vertex $v_i \in U \cup W$ has an associated *capacity* $c_i \geq 1$. We refer to $G$ as a *capacitated bipartite graph* and a *matching* $M$ of $G$ is a subset of $E$ such that for each $v_i \in U \cup W$, $|e \in M : v_i \in e| \leq c_i$. Note that in this thesis, we are concerned only with capacitated bipartite graphs in which the vertices in $U$ have a capacity equal to 1 (and the vertices in $W$ can have non-unitary capacity). The problem of finding a maximum matching in $G$ is also referred to in the literature as the *maximum cardinality degree-constrained subgraph problem* or maximum cardinality DCS in short, and Gabow's algorithm [15] provides the fastest way to solve this, taking $O(\sqrt{C}m)$ time, where $C = \sum_{j=1}^{n_2} c_j$ denote the sum of the capacities of the vertices in $W$.

### 1.2.2 Weighted Graphs

Let $G = (U, W, E)$ be a bipartite graph where each edge $e \in E$ has an associated *weight* $wt(e) \in \mathbb{N}$. We define the *weight* of a matching $M$ of $G$ as $wt(M) = \sum_{e \in M} wt(e)$. A common problem, given any weighted bipartite graph $G$, is to find a maximum weight matching of $G$. This is also known as the *Assignment problem* [46].

For the case where all vertices have capacity 1, the running time of the best algorithm is usually stated as $O(nm + n^2 \log n)$ [14]. For the capacitated bipartite graph, the fastest time to solve the problem is due to Gabow's algorithm for the *maximum weight degree-constrained subgraph problem*, or maximum weight DCS in short, which takes $O(C \min(m \log n, n^2))$ time [15].

### 1.2.3 Edmonds-Gallai Decomposition

Let $G$ be some bipartite graph where all the vertices have capacity 1. The Edmonds-Gallai Decomposition (see [4, 35, 47]) is a well-known result in matching theory that gives an important characterisation of maximum matchings in $G$. That is, let $M$ be a maximum matching in $G$. Then, the vertices of $G$ can be partitioned into three disjoint sets: $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$. Vertices in $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ are called *even*, *odd* and *unreachable* respectively. A vertex $v$ is even (odd) if there exists an alternating path of even (odd) length from a vertex that is unmatched in $M$ to $v$. If no such alternating path exists, $v$ is unreachable. We henceforth refer to this vertex labelling as an *EOU labelling*. The fundamental results of the Edmonds-Gallai Decomposition are summarised in the following lemma, the proof of which can be obtained explicitly from [4].

**Lemma 1.2.1.** *Let $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ be the vertex sets defined by $G$ and $M$ above. Then,*

*(a) The sets $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ are pairwise disjoint. Every maximum matching in $G$ partitions the vertices into the same sets of even, odd and unreachable vertices.*

*(b) Every maximum matching $M$ in $G$ satisfies the following properties:*

*(i) every vertex in $\mathcal{O}$ and every vertex in $\mathcal{U}$ is matched;*

*(ii) every vertex in $\mathcal{O}$ is matched to a vertex in $\mathcal{E}$;*

*(iii) every vertex in $\mathcal{U}$ is matched to another in $\mathcal{U}$;*

*(iv) $|M| = |\mathcal{O}| + |\mathcal{U}|/2$.*

*(c) No maximum matching in $G$ contains an edge between two vertices in $\mathcal{O}$ or a vertex in $\mathcal{O}$ with a vertex in $\mathcal{U}$. There is no edge in $G$ connecting a vertex in $\mathcal{E}$ with a vertex in $\mathcal{U}$, or between two vertices of $\mathcal{E}$.*

## 1.3 Bipartite matching problems, one-sided preference lists

### 1.3.1 House Allocation Problem

Bipartite matching problems involving two sets of participants, namely a set of agents and a set of objects, are commonly referred to as House Allocation problems [1–3, 16, 19, 24]. These problems have been widely studied not only due to their theoretical interest, but also in view of their practical importance. Widespread applications occur in real-life resource allocation problems such as campus housing allocation in US universities [1], hence the problem name; in assigning probationary teachers to their first posts in Scotland; and in Amazon's DVD rental service.

An instance $I$ of the House Allocation problem (HA) comprises two disjoint sets $A$ and $H$, where $A = \{a_1, a_2, ..., a_{n_1}\}$ is the set of *agents* and $H = \{h_1, h_2, ..., h_{n_2}\}$ is the set of *houses*. Each agent $a_i \in A$ ranks in strict order a subset of those houses in $H$ giving rise to his *preference list*. If $a_i$ ranks a house $h_j \in H$ in his preference list, we say that $a_i$ finds $h_j$ *acceptable*. An agent $a_i$ *prefers* one house $h_j$ to another house $h_l$ if $h_j$ precedes $h_l$ in $a_i$'s preference list. We define the *underlying graph* of $I$ to be the bipartite graph $G = (A, H, E)$, where $E$ is the set of edges in $G$ representing the acceptable houses of the agents. We let $n = n_1 + n_2$ and $m = |E|$.

Given an agent $a_i \in A$ and an acceptable house $h_j \in H$ for $a_i$, we define $rank_{a_i}(h_j)$ to be the number of agents that $a_i$ prefers to $h_j$ plus 1. If $rank_{a_i}(h_j) = k$, we say that $h_j$ is the *kth choice* of $a_i$. Let $(a_i, h_j) \in E$ be any edge. Then, we define the rank of $(a_i, h_j)$ to be $r(a_i, h_j) = rank_{a_i}(h_j)$. Let $z \leq n_2$ be the maximum length of any agent's preference list in $I$. Clearly, $z$ corresponds to the largest rank of a house taken over all agents' preference lists in $I$. We assume that no agent has an empty preference list and each house is acceptable to at least one agent, i.e., $m \geq \max\{n_1, n_2\}$.

An *assignment* $M$ is a subset of $A \times H$ such that $(a_i, h_j) \in M$ only if $a_i$ finds $h_j$ acceptable. If $(a_i, h_j) \in M$, we say that $a_i$ and $h_j$ are *assigned* to each other, and we call $a_i$ and $h_j$ *partners* in $M$. A *matching* is an assignment $M$ such that (i) each agent is assigned to at most one house in $M$, and (ii) each house $h_j \in H$ is assigned to at most one agent in $M$. If a participant $p \in A \cup H$ is assigned in $M$, we denote by $M(p)$ the participant that $p$ is assigned to in $M$. If $p$ is not assigned in $M$, we say that $p$ is *unassigned*. Given two matchings $M$ and $M'$ in $G$, we say that an agent $a_i$ *prefers* $M'$ to $M$ if either (i) $a_i$ is assigned in $M'$ and unassigned in $M$, or (ii) $a_i$ is assigned in both $M'$ and $M$ and prefers

$M'(a_i)$ to $M(a_i)$. We use $\mathcal{M}$ to denote the set of all matchings in $I$.

Several variants of HA may be formulated as follows.

#### 1.3.1.1 Variants of HA

First of all, we can have a straightforward extension of HA by allowing *ties* in the agents' preference lists. A tie between two houses $h_j$ and $h_l$ occurs in an agent $a_i$'s preference list when $rank_{a_i}(h_j) = rank_{a_i}(h_l)$, and we say that the agent $a_i$ is *indifferent* between $h_j$ and $h_l$. The problem then becomes known as the House Allocation problem with Ties, or HAT for short.

We can have a variant of HA in which each agent $a$ has an assigned positive weight $w(a)$ that indicates his priority (which may be based on such objective criteria relevant to the matching application). This is known as the Weighted House Allocation problem, denoted by WHA, or WHAT if ties are present. If the houses are allowed to have non-unitary capacity, we then have a generalisation of HA that is known as the Capacitated House Allocation problem, denoted by CHA, or CHAT if ties are present. A third possible variant of HA combines WHA and CHA by letting the agents have a positive weight to indicate their priority, and allowing the houses to have non-unitary capacity. This is known as the Weighted Capacitated House Allocation problem, denoted by WCHA, or WCHAT if ties are present.

We remark that all the notations and terminology that were defined for HA in Section 1.3.1 carry over directly to each of its variants with the exception of some terms that we will require to define separately. We henceforth assume these definitions in any variant of HA in the rest of this thesis and explicitly define relevant concepts only where we need to adapt them to the context of the variant.

In each of CHA, CHAT, WCHA and WCHAT, we require to redefine a matching since each house $h_j$ may now have a non-unitary capacity $c_j \geq 1$, and a many-one mapping of the agents and houses is sought in these contexts instead. Here, we define a *matching* to be an assignment $M$ such that (i) each agent is assigned to at most one house in $M$, and (ii) each house $h_j \in H$ is assigned to at most $c_j$ agents in $M$. Consequently, $M(h_j)$ refers to the set of agents assigned to $h_j$ in $M$ (which could be empty) in these contexts. If $|M(h_j)| < c_j$, we say that $h_j$ is *undersubscribed* in $M$; otherwise $h_j$ is *full* in $M$. We also let $C = \sum_{j=1}^{n_2} c_j$ denote the sum of the capacities of the houses.

### 1.3.2 Optimality criteria for bipartite matching problems

For bipartite matching problems with one-sided preferences, various criteria as to what constitutes an "optimal" matching have been considered. In this section, we give a review of the optimality criteria that are considered in this thesis.

#### 1.3.2.1 Pareto optimal matchings

One solution concept that has received much attention, particularly from the Economics community is *Pareto optimality* [1–3, 7, 51, 56, 57], because it is regarded by Economists as a fundamental property to be satisfied in the context of matching problems. Let $I$ be an instance of HA or any or its variants. Then, we may define a relation $\prec$ on $\mathcal{M}$ based on the preference of agents over matchings in $I$ (as defined above): that is, given any two matchings $M$ and $M'$, $M' \prec M$ if and only if no agent prefers $M$ to $M'$, and some agent prefers $M'$ to $M$. A matching $M$ is defined to be *Pareto optimal* if and only if there is no other matching $M'$ such that $M' \prec M$.

Various algorithms exist for finding a Pareto optimal matching in any given instance of HA, the most straightforward being a greedy algorithm known as the *serial dictatorship* mechanism [1, 56] which considers each agent $a$ in turn, and gives $a$ his most preferred vacant house (if such a house exists). However, such an algorithm may fail to find a Pareto optimal matching of maximum cardinality (henceforth a maximum Pareto optimal matching), which is undesirable in applications that seek to assign as many agents as possible (see Chapter 2 for further details).

Abraham et al. [3] gives the fastest algorithm, which takes $O(\sqrt{n}m)$ time, for finding a maximum Pareto optimal matching given an HA instance. In this thesis, we extend their results to the capacitated bipartite graph case in Chapter 2 by constructing an $O(\sqrt{C}m)$ time algorithm for finding a maximum Pareto optimal matching given any instance of CHA. Since the definition of a Pareto optimal matching in WCHA is identical to that in CHA, this algorithm can also be used for the analogous problem in the weighted capacitated bipartite graph case.

#### 1.3.2.2 Popular matchings

Another important solution concept is that of a *popular matching*. Let $I$ be an instance of CHAT. Also, let $M$ and $M'$ be two arbitrary matchings in $I$ and let $P(M, M')$ denote the set of agents who prefer $M$ to $M'$. We say that $M$ is *more popular than* $M'$ if

$|P(M, M')| > |P(M', M)|$, i.e. the number of agents who prefer $M$ to $M'$ is greater than the number of agents who prefer $M'$ to $M$. A matching $M$ in $I$ is *popular* if there is no other matching $M'$ in $I$ that is more popular than $M$.

We remark that the definition of a popular matching can be extended to WCHAT in the following way. First of all, given any two matchings $M$ and $M'$ in a weighted setting, we define the *satisfaction* of $M$ with respect to $M'$ to be $sat(M, M') = \sum_{a \in P(M,M')} w(a) - \sum_{a \in P(M',M)} w(a)$. We then say that $M$ is *more popular than* $M'$ if $sat(M, M') > 0$. A matching $M$ is defined to be *popular* if there is no other matching in the problem instance that is more popular than $M$.

Gärdenfors [20] first introduced the notion of a popular matching (referring to this concept as a *majority assignment*) in the context of voting theory. We remark that the *more popular than* concept can be traced back even further to the Condorcet voting protocol. Popular matchings were then considered by Abraham et al. [4] in the context of HA. They showed that popular matchings need not exist, given an instance of HA, and also noted that popular matchings can have different cardinalities. The same authors described an $O(n + m)$ algorithm for finding a maximum cardinality popular matching (henceforth a maximum popular matching) if one exists, given an instance of HA. They also described an $O(\sqrt{n}m)$ counterpart for HAT.

Mahdian [37] showed that a popular matching exists with high probability given an instance of HAT when (i) preference lists are random, and (ii) the number of houses is a small multiplicative factor larger than the number of agents. To cope with the possible non-existence of a popular matching, McCutchen [40] defined two notions of a matching that are, in some sense, "as popular as possible", namely a *least-unpopularity-factor matching* and a *least-unpopularity-margin matching*. McCutchen proved that computing either type of matching is NP-hard. Abraham and Kavitha [5] considered *voting paths* in relation to popular matchings in a dynamic matching market in which agents and houses can enter and leave the market. Mestre [43] then described an $O(n + m)$ algorithm for finding a maximum popular matching if one exists, given an instance of WHA. He also described an $O(\min(k\sqrt{n}, n)m)$ counterpart for WHAT, where $k$ is the maximum priority of any agent.

In Chapter 4, we consider popular matchings in CHAT. For the case where preference lists are strict, we give an $O(\sqrt{C}n_1 + m)$ time algorithm to find a maximum popular matching, or to report that none exists. We then show how to extend Lemma 1.2.1, an

important result in matching theory, to the capacitated bipartite graph case, and use this result to construct an $O(\sqrt{C}m)$ time algorithm to find a maximum popular matching or report that none exists in a given CHAT instance. We also consider the analogous problem in WCHA in Chapter 5. There, we identify a structure in the underlying graph of the problem that singles out those edges that cannot belong to a popular matching. We then use this to construct a $O(\sqrt{C}n_1 + m)$ time algorithm that finds a maximum popular matching, or reports that none exists in a given WCHA instance.

#### 1.3.2.3 Profile-based optimal matchings

Finally, let $I$ be an instance of HAT or any of its variants. Recall that $z$ is the largest rank of a house taken over all agents' preference lists in $I$. Define the *profile* $\rho(M)$ of a matching $M$ in $I$ to be the $z$-tuple $(x_1, x_2, ..., x_z)$ where for each $i$ $(1 \leq i \leq z)$, $x_i$ is the number of agents who are assigned in $M$ with one of their $i$th choice houses. Then, it is possible to define at least three versions of what it means for a matching to be optimal based on its profile.

Informally, a *greedy maximum* matching is a matching that has lexicographically maximum profile taken over all maximum matchings. On the other hand, a *rank-maximal* matching is a matching that has lexicographically maximum profile taken over all matchings. Finally, a *generous maximum* matching is a matching whose reverse profile is lexicographically minimum taken over all maximum matchings. We remark that each of a rank-maximal, a greedy maximum and a generous maximum matching must be Pareto optimal; however, they are not necessarily popular.

The fastest combinatorial approach for finding a rank-maximal matching given an HAT instance is described by Irving et al. [29], and this takes $O(\min(z^*\sqrt{n}, n + z^*)m)$ time where $z^*$ is the maximal rank of an edge in an optimal solution. Kavitha and Shah [33] studied rank-maximal matchings in WHAT and described an $O(\min(z^*\sqrt{n}, n+z^*)m)$ time algorithm for solving the problem. In an unpublished manuscript [28], Irving describes an approach based on the Bellman-Ford algorithm to find greedy maximum and generous maximum matchings in HAT.

In Chapter 7, we consider the individual problems of finding a rank-maximal, a greedy maximum and a generous maximum matching in the context of CHAT. For the case of finding a rank-maximal matching, we construct an $O(\min(z^*\sqrt{C}, C+z^*)m)$ time algorithm for solving the problem. For each of the cases of finding a greedy maximum and a generous

maximum matching, we explore two alternative solutions for the problem, the faster of which (in most practical applications as we shall show) takes $O(zC \min(m \log n, n^2))$ time.

## 1.4 Bipartite matching problems, two-sided preference lists

### 1.4.1 One-one mapping: the classical Stable Marriage Problem

The classical Stable Marriage problem (SM) is a widely studied example of a combinatorial problem in the category indicated by this subsection. An instance $I$ of SM involves two disjoint sets $U$ and $W$ where $U = \{u_1, u_2, ..., u_n\}$ is the set of men, and $W = \{w_1, w_2, ..., w_n\}$ is the set of women. Each person $p \in U \cup W$ ranks all members of the opposite sex in strict order of preference giving rise to his/her *preference list*. We say that person $p$ *prefers* $q$ to $r$ if $q$ precedes $r$ on $p$'s preference list.

An *assignment* $M$ is a subset of $U \times W$ such that $(u_i, w_j) \in M$ only if $u_i$ and $w_j$ find each other acceptable. If $(u_i, w_j) \in M$, we say that $u_i$ and $w_j$ are *assigned* to each other. A *matching* in $I$ is an assignment $M$ such that (i) each man is assigned to at most one woman in $M$, and (ii) each woman is assigned to at most one man in $M$. If $(u_i, w_j) \in M$, $u_i$ and $w_j$ are called *partners* in $M$. A *blocking pair* for $M$ is a (man,woman) pair $(u_i, w_j)$ such that $u_i$ prefers $w_j$ to $M(u_i)$ and $w_j$ prefers $u_i$ to $M(w_j)$, where $M(q)$ denotes $q$'s partner in $M$ for any person $q$ in $I$. A matching that admits no blocking pair is said to be *stable*.

Stable matching problems were first studied by Gale and Shapley [17] in their seminal paper "College Admissions and the Stability of Marriage". There they gave an algorithm, now widely known as the Gale-Shapley (GS) algorithm, that always finds a stable matching for any instance of SM in $O(n^2)$ time [34]. Very briefly, the algorithm involves a sequence of "proposals" from members of one sex to members of the opposite sex and it terminates when everyone becomes engaged. If the men were the proposers, then we obtain the *man-oriented* version of the GS algorithm, otherwise the algorithm is known as *woman-oriented*. The algorithm is inherently non-deterministic in that the order in which the proposals take place is of no consequence to the result [22].

Gale and Shapley [17] observed that the man-oriented version of the GS algorithm always gives the *man-optimal* stable matching, in which each man has the best partner that he can have in any stable matching. The man-optimal stable matching is also *woman-pessimal*, for each woman has the worst partner that she can have in any stable matching

[41]. If the woman-oriented version of the GS algorithm is used, then this gives analogous results: we obtain the *woman-optimal* stable matching which is *man-pessimal*. Gusfield and Irving [22] gave an extended version of the man-oriented GS algorithm which simplifies the process by deleting from a woman $w$'s preference list every man $u'$ who succeeds a man $u$ from whom she has received a proposal. This is because no such pair $(u', w)$ can be part of any stable matching.

Several variants of the Stable Marriage problem exist and have been widely studied as follows.

#### 1.4.1.1 Incomplete lists

A natural variant of SM occurs when each person $p$ in an SM instance $I$ need not rank all members of the opposite sex. Then the preference list for each person $p$ contains a subset of members of the opposite sex such that person $p$ finds $q$ *acceptable* if and only if $q$ appears in $p$'s preference list. We henceforth assume in all contexts where all the participants are agents, that if an agent $a$ ranks another agent $b$ in $a$'s preference list, then $b$ also ranks $a$ in $b$'s preference list. Furthermore, the numbers of men and women need not be equal. We say that these preference lists are *incomplete* and use SMI (Stable Marriage with Incomplete Lists) to denote this version of SM.

In this setting, a man $u_i$ and a woman $w_j$ are assigned to each other in a matching $M$ only if $u_i$ and $w_j$ are acceptable to one another. Thus, matchings need not be complete, i.e. not all members of either sex need be assigned in a given matching in this setting. Here, a (man,woman) pair $(u_i, w_j)$ constitutes a blocking pair for $M$ whenever (i) $u_i$ and $w_j$ find each other acceptable, (ii) $u_i$ is either unassigned in $M$ or prefers $w_j$ to $M(u_i)$, and (iii) $w_j$ is either unassigned in $M$ or prefers $u_i$ to $M(w_j)$. A matching in an instance of SMI is *stable* if it admits no such blocking pair. Every SMI instance admits a stable matching [17], and Gusfield and Irving [22] showed that the extended GS algorithm can be used to find a stable matching, given an SMI instance. Furthermore, for any matching $M$ in an instance of SMI, some agents may be unassigned in $M$, but the same agents are unassigned in all stable matchings and as a consequence, all stable matchings in $I$ have the same cardinality [18].

#### 1.4.1.2 Ties

Another variant of SM occurs when the preference list of each person is allowed to contain ties. We say that a person $p$ is *indifferent* between $q$ and $r$ if $q$ and $r$ appear in a tie in $p$'s preference list, and use SMT (Stable Marriage with Ties) to denote this variant of SM. The introduction of ties in a person's preference list gives rise to three definitions of stability, namely *weak stability*, *strong stability* and *super-stability* [26].

A matching $M$ is defined to be *weakly stable* if there does not exist any blocking pair $(u_i, w_j)$ such that $u_i$ and $w_j$ prefer each other to their partners in $M$. On the other hand, a matching $M$ is *strongly stable* if there does not exist any blocking pair $(u_i, w_j)$ such that either (i) $u_i$ prefers $w_j$ to $M(u_i)$, and $w_j$ either prefers $u_i$ to $M(w_j)$ or is indifferent between them, or (ii) $w_j$ prefers $u_i$ to $M(w_j)$, and $u_i$ either prefers $w_j$ to $M(u_i)$ or is indifferent between them. We define a matching $M$ to be *super-stable* if there does not exist any blocking pair $(u_i, w_j)$ such that $u_i$ either prefers $w_j$ to $M(u_i)$ or is indifferent between them, and $w_j$ either prefers $u_i$ to $M(w_j)$ or is indifferent between them.

A weakly stable matching can always be found for an instance of SMT by simply breaking the ties arbitrarily and then applying the extended Gale-Shapley algorithm to the derived instance. This guarantees to produce a matching that is weakly stable in the original instance with ties [22]. Also, all weakly stable matchings have the same cardinality in this context. We remark that strongly stable matchings and super-stable matchings need not exist for a given instance of SMT; hence, we do not devote any more attention to the results concerning these versions of stability and refer the reader to [26] for more details.

#### 1.4.1.3 Ties and Incomplete lists

SMT and SMI can be combined to give the Stable Marriage problem with Ties and Incomplete lists, or SMTI in short. That is, a given preference list in SMTI can be incomplete and can contain ties. In addition, the definition of weak stability can be extended from SMT to SMTI in a natural way. A weakly stable matching may be found using the same algorithm described for the corresponding problem in SMT. Unlike the case in SMT, weakly stable matchings can have different cardinalities, and Manlove et al. [39] shows that the problem of finding a maximum cardinality weakly stable matching given an instance of SMTI is NP-hard, even if the ties are at the tails of the lists and on one side only, there is at most one tie per list, and each tie is of length two.

### 1.4.2 One-many mapping: the Hospitals-Residents Problem

The Hospitals-Residents problem is a many-one extension of SM that was first considered by Gale and Shapley [17] and referred to in that paper as the College Admissions problem. This problem has since invariably been known as the Hospitals-Residents problem mainly because of its applications in the medical matching context such as the SFAS and NRMP as mentioned in Section 1.1.

An instance $I$ of the Hospitals-Residents problem (HR) comprises two disjoint sets $R$ and $H$, 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. Each resident $r_i \in R$ ranks a subset of the hospitals in $H$ in strict order of preference giving rise to his *preference list*. Similarly, each hospital $h_j \in H$ ranks a subset of the residents in $R$ in strict order, giving rise to its preference list. If $r_i$ and $h_j$ rank each other in their preference lists, we say that they find each other *acceptable*, and $r_i$ and $h_j$ are each an *acceptable partner* for one another. We say that a resident $r_i$ *prefers* one hospital $h_j$ to another $h_k$ if $h_j$ precedes $h_k$ in $r_i$'s preference list. Similarly, we define the preferences of hospitals over residents. Each hospital $h_j \in H$ has a *capacity* $c_j$ which indicates the maximum number of posts it may fill. We define the *underlying graph* of $I$ to be the bipartite graph $G = (R, H, E)$, where $E$ is the set of edges in $G$ representing the acceptable hospitals of the residents. Let $C = \sum_{j=1}^{n_2} c_j$ denote the sum of the capacities of the hospitals. We also let $n = n_1 + n_2$ and $m = |E|$.

An *assignment* $M$ is a subset of $R \times H$ such that $(r_i, h_j) \in M$ only if $r_i$ finds $h_j$ acceptable and vice versa. If $(r_i, h_j) \in M$, we say that $r_i$ and $h_j$ are *assigned* to each other. A *matching* in $I$ is an assignment $M$ such that (i) each resident is assigned to at most one hospital in $M$, and (ii) each hospital $h_j \in H$ is assigned to at most $c_j$ residents in $M$. If a resident $r_i \in R$ is assigned in $M$, we denote by $M(r_i)$ the hospital that $r_i$ is assigned to in $M$. We define $M(h_j)$ to be the set of residents assigned to $h_j$ in $M$ (thus $M(h_j)$ could be empty). We say that a hospital $h_j \in H$ is *full* in $M$ if $|M(h_j)| = c_j$, and *undersubscribed* in $M$ if $|M(h_j)| < c_j$.

A *blocking pair* for $M$ is a (resident,hospital) pair $(r_i, h_j)$ such that

- $r_i$ and $h_j$ find each other acceptable
- either $r_i$ is unassigned in $M$, or $r_i$ prefers $h_j$ to $M(r_i)$
- either $h_j$ is undersubscribed in $M$, or $h_j$ prefers $r_i$ to its worst assigned resident in $M(h_j)$

A matching that admits no blocking pair is said to be *stable*, and every instance of HR admits a stable matching [22]. Note that SMI is a special case of HR in which $c_j = 1$ for all $h_j \in H$. Furthermore, we can extend the definition of a man-optimal and a woman-optimal stable matching in SMI to a *resident-optimal* and a *hospital-optimal* stable matching respectively in HR (see Section 1.6 of [22]). For any given instance $I$ of HR, efficient algorithms exist to find such stable matchings of $I$ [22]. An HR instance can have more than one stable matching. However, all stable matchings have the same cardinality, and the same residents are assigned in all stable matchings [18, 49]. Furthermore, any hospital that is undersubscribed in one stable matching is assigned with exactly the same residents in all stable matchings [50]. Collectively, these results are known as the *Rural Hospitals Theorem* because of their historical significance relating to the problems that rural hospitals face when recruiting interns in the NRMP [22].

Given two matchings $M$ and $M'$, we say that a resident $r_i$ *prefers* $M'$ to $M$ if either (i) $r_i$ is assigned in $M'$ and unassigned in $M$, or (ii) $r_i$ is assigned in both $M'$ and $M$ and prefers $M'(r_i)$ to $M(r_i)$. Unlike the case for residents, it is less straightforward to define the preference of a hospital $h_j$ over two matchings since $h_j$ may have non-unitary capacity. Given that the primary goal of many practical matching applications is to maximise the number of agents assigned, as well as to optimise the satisfaction of the agents according to their preference lists, we give what may be viewed as a definition of a hospital $h_j$'s preference over matchings in $I$ as follows.

**Definition 1.4.1.** *We say that the hospital* $h_j$ prefers *one matching* $M'$ *to another* $M$ *if*

1. $|M'(h_j)| > |M(h_j)|$, *or*
2. $|M'(h_j)| = |M(h_j)|$ *and* $h_j$ *prefers the worst resident assigned to it in* $M'$ *to the worst resident assigned to it in* $M$.

Note that even though there are no ties in $h_j$'s preference list, Definition 1.4.1 allows a hospital $h_j$ to be indifferent between two matchings $M$ and $M'$ if $|M(h_j)| = |M'(h_j)|$, the worst resident assigned to $h_j$ is the same in both $M$ and $M'$ but $h_j$ has different sets of residents assigned to it in $M$ and $M'$. If $h_j$ does not prefer $M'$ to $M$, and also does not prefer $M$ to $M'$, we say that $h_j$ is *indifferent* between $M$ and $M'$.

As is the case in SMI, we can permit ties in the preference lists in this context, and use HRT (Hospital-Residents problem with Ties) to denote this variant of HR. The definition of weak stability carries over from SMTI to HRT in an analogous way to the extension of

the definition of classical stability from SMI to HR. Since SMTI is a special case of HRT, it follows that the problem of finding a maximum cardinality weakly stable matching is also NP-hard in HRT.

We remark that each of the concepts of a Pareto optimal matching, a popular matching and a profile-based optimal matching, can be defined in SM, HR, and their respective variants in the same way as the respective concepts were defined in the context of HA and its variants in Section 1.3.2. Given that stable matchings sometimes do not satisfy the key requirement in many practical matching contexts, which is to maximise the number of agents assigned in any given matching (as we shall show), we thus also apply these optimality criteria to SM and HR and some of their variants, and obtain new results as follows.

In Chapter 3, we study the problem of finding a maximum Pareto optimal matching given an instance of HR, and describe an $O(\sqrt{C}m)$ time algorithm for its solution. We also show how this algorithm can be adapted to solve the analogous problem given an instance of SMI in $O(\sqrt{n}m)$ time. We then consider the structure of popular matchings in SMTI-SYM, a special case of SMTI where preference lists are symmetric, in Chapter 6. Little is known about how to find a maximum popular matching, or to determine that none exists, in the bipartite setting where all the participants are agents (i.e. all participants have preferences). A first step in this direction is presented by our characterisation results of popular matchings in SMTI-SYM in Chapter 6. There, we also give an $O(\sqrt{n}m)$ time algorithm for testing if a matching in a given SMTI-SYM instance is popular. We also consider the individual problems of finding a rank-maximal, a greedy maximum and a generous maximum matching in the context of HRT in Chapter 8. For each problem, we explore two alternative algorithms for its solution, the faster of which (in most practical applications as we shall show) takes $O(zC \min(m \log n, n^2))$ time. We also show how this algorithm can be adapted to solve the analogous problem given an instance of SMTI in $O(z(nm + n^2 \log n))$ time.

## 1.5 Non-bipartite matching problems

### 1.5.1 Stable Roommates Problem

In an instance of the Stable Roommates (SR) problem, first introduced by Gale and Shapley [17], there is a set of $n$ agents where $n$ is even. Each agent ranks the $n-1$ others

in strict order of preference. A *matching* $M$ is a partition of the set of agents into disjoint pairs. A *blocking pair* for $M$ is a pair of agents $\{x, y\} \notin M$ such that $x$ prefers $y$ to $M(x)$ and $y$ prefers $x$ to $M(y)$ where $M(q)$ denotes $q$'s partner in $M$ for any agent $q$. A matching is *stable* if it admits no blocking pair.

It is well-known that SM is just a special case of SR, since the set of stable matchings is unchanged if we reduce an SM instance $I$ into an SR instance by appending to the very end of each agent's preference list all the other agents that are of the same sex in $I$ [22]. Not all SR instances admit a stable matching [17], and Knuth [34] posed the question of whether the problem of determining the solvability of SR instances might be NP-complete. This question was answered by Irving [25], who gave an $O(n^2)$ algorithm for finding a stable matching or reporting that no such matching exists. Alternative approaches for finding a stable matching if one exists, given an SR instance have since been described [12,13,58–60].

As with SM, we may formulate an extension of SR where preference lists may include ties and be incomplete (SRTI). In such a setting, the definition of a weakly stable matching may be extended from the SMTI context in a natural way given an SRTI instance, and weakly stable matchings, if they exist, can have different cardinalities. The problem of finding a maximum cardinality weakly stable matching given an SRTI instance is NP-hard [30, 48].

We remark that, as in SM, HR and their respective variants, each of the concepts of a Pareto optimal matching, a popular matching and a profile-based optimal matching, can be defined similarly in SR and its variants as they were defined in Section 1.3.2. Pareto optimal matchings in SR was recently studied by Abraham and Manlove [7]. There, the authors gave an $O(\sqrt{n\alpha(m,n)}m\log^{3/2}n)$ time algorithm for the problem of finding a maximum Pareto optimal matching in an SR instance $I$, where $n$ is the number of agents, $m$ is the total length of the preference lists in $I$ and $\alpha$ is the inverse Ackermann function. Chung [10] considered popular matchings in instances of SR and noted that a stable matching is popular; however, the same need not be true in the presence of ties. Abraham et al. [6] studied rank-maximal matchings in a special case of SR in which roommate pairs are ranked globally, and gave an $O(\min(z^*\sqrt{n}, z^*+n)m)$ time algorithm for the solution to the problem. Little is known about the individual problems of finding a popular matching (if one exists) and finding profile-based optimal matchings, given the general case of SR.

# Chapter 2

# Pareto optimal matchings in CHA

## 2.1 Introduction

As mentioned in Section 1.3.2, Pareto optimality is a solution concept that has received much attention from the Economics community in the context of matching problems since it is regarded as a fundamental solution concept. Pareto optimality interests us from the point of view of this research because most of the associated algorithmic questions have not, on the other hand, been considered extensively in the literature.

In this chapter, we study the problem of finding a maximum Pareto optimal matching in the context of CHA, a general case of bipartite matching problems with one-sided preferences. The main results of this chapter, and their organisation are as follows. We give some terminology and preliminary results on Pareto optimal matchings in CHA in Section 2.2. We then give a characterisation of Pareto optimal matchings in CHA in Section 2.3, which we subsequently use in Section 2.4 to construct an $O(\sqrt{C}m)$ time algorithm for finding a maximum Pareto optimal matching given an instance $I$ of CHA where $C$ is the total capacity of the houses and $m$ is the total length of preference lists in $I$ respectively. Note that we reuse most of the terminology and notation from HA as defined in Section 1.3.1, and we explicitly define relevant concepts only where we need to adapt them to CHA.

## 2.2 Basic terminology and preliminary results

Let $I$ be an instance of CHA, and let $G = (A, H, E)$ be the underlying bipartite graph of $I$ as defined in Section 1.3.1. Each house $h_j \in H$ has a *capacity* $c_j \geq 1$ which indicates the

maximum number of agents that may be assigned to it. Recall from Section 1.3.1 that an *assignment* $M$ is a subset of $A \times H$ such that $(a_i, h_j) \in M$ only if $a_i$ finds $h_j$ acceptable. Furthermore, if $(a_i, h_j) \in M$, we say that $a_i$ and $h_j$ are *assigned* to each other, and we call $a_i$ and $h_j$ *partners* in $M$. A *matching* $M$ in an instance $I$ of CHA is an assignment such that (i) each agent is assigned to at most one house in $M$, and (ii) each house $h_j \in H$ is assigned to at most $c_j$ agents in $M$. If an agent $a_i \in A$ is assigned in $M$, we denote by $M(a_i)$ the house that $a_i$ is assigned to in $M$. We define $M(h_j)$ to be the set of agents assigned to $h_j$ in $M$ (thus $M(h_j)$ could be empty). We say that a house $h_j \in H$ is *full* in $M$ if $|M(h_j)| = c_j$, and *undersubscribed* in $M$ if $|M(h_j)| < c_j$. We assume that no agent has an empty preference list and each house is acceptable to at least one agent so that $m \geq \max\{n_1, n_2\}$. Let $C = \sum_{j=1}^{n_2} c_j$ denote the sum of the capacities of the houses.

## 2.3 Characterisation of Pareto optimal matchings

Let $M$ be a matching in $I$. We say that $M$ is *maximal* if there is no agent $a_i \in A$ and house $h_j \in H$ such that $a_i$ is unassigned in $M$, $h_j$ is undersubscribed in $M$ and $a_i$ finds $h_j$ acceptable. Also, $M$ is *trade-in-free* if there is no (agent,house) pair $(a_i, h_j)$ such that $a_i$ is assigned in $M$, $h_j$ is undersubscribed in $M$ and $a_i$ prefers $h_j$ to $M(a_i)$.

A *cyclic coalition* with respect to $M$ is a sequence of distinct assigned agents $C = \langle a_0, a_1, \ldots, a_{r-1} \rangle$, for some $r \geq 2$, such that $a_i$ prefers $M(a_{i+1})$ to $M(a_i)$ for each $i$ ($0 \leq i \leq r-1$). Henceforth, all subscripts are taken modulo $r$ when reasoning about coalitions.

Given a cyclic coalition $C$, the matching

$$M' = (M \backslash \{(a_i, M(a_i)) : 0 \leq i \leq r-1\}) \cup \{(a_i, M(a_{i+1})) : 0 \leq i \leq r-1\}$$

is defined to be the matching obtained from $M$ by *satisfying* $C$. We say that $M$ is *cyclic-coalition-free* if $M$ admits no cyclic coalition. The following lemma gives a necessary and sufficient condition for a matching to be Pareto optimal.

**Lemma 2.3.1.** *Let $M$ be a matching in a given instance $I$ of* CHA*. Then $M$ is Pareto optimal if and only if $M$ is maximal, trade-in-free and cyclic-coalition-free.*

*Proof.* Let $M$ be a Pareto optimal matching. Suppose for a contradiction that $M$ is not maximal. It follows that there exist an agent $a_i$ and a house $h_j$ such that $a_i$ is unassigned in $M$, $h_j$ is undersubscribed in $M$ and $a_i$ finds $h_j$ acceptable. Let $M' = M \cup \{(a_i, h_j)\}$. Then, $M' \prec M$, a contradiction. Now, suppose for a contradiction that $M$ is not trade-in-free.

It follows that there exist an agent $a_i$ and a house $h_j$ such that $a_i$ is assigned in $M$, $h_j$ is undersubscribed in $M$, and $a_i$ prefers $h_j$ to $M(a_i)$. Let $M' = (M \setminus \{(a_i, M(a_i))\} \cup \{(a_i, h_j)\}$. Then, $M' \prec M$, a contradiction. Finally, suppose that $M$ admits some cyclic coalition $C$. Let $M'$ be the matching obtained by satisfying $C$. Clearly then, $M' \prec M$, a contradiction.

Conversely, let $M$ be a matching that is maximal, trade-in-free and cyclic-coalition-free. Let us suppose for a contradiction that $M$ is not Pareto optimal. Then there exists some matching $M'$ such that $M' \prec M$. Let $G$ be the underlying graph of $I$. We clone $G$ to obtain a cloned graph $C(G)$ as follows. We replace every house $h_j \in H$ with the clones $h_j^1, h_j^2, \ldots, h_j^{c_j}$. We then divide the capacity of each house among its clones by allowing each clone to have capacity 1. In addition, if $(a_i, h_j) \in G$, then we add $(a_i, h_j^p)$ to $C(G)$ for all $p$ $(1 \leq p \leq c_j)$. Let us then adapt the matching $M$ in $G$ to obtain its clone $C(M)$ in $C(G)$ as follows. If a house $h_j$ in $G$ is assigned to $x_j$ agents $a_1, ...a_{x_j}$ in $M$, then we add $(a_p, h_j^p)$ to $C(M)$ for $1 \leq p \leq x_j$, so that $|C(M)| = |M|$. We repeat a similar process for $M'$ to obtain its clone $C(M')$ in $C(G)$.

Let us consider $X = C(M) \oplus C(M')$ and let $C$ be a connected component of $X$. It follows that $C$ is a path or cycle whose edges alternate between $C(M)$ and $C(M')$. Now, $C$ cannot be an even-length alternating path that has more agents than houses or an an odd-length alternating path whose end edges are in $C(M)$, for otherwise we have an agent who is assigned in $M$ but unassigned in $M'$, a contradiction since $M' \prec M$. In addition, $C$ cannot be an even-length alternating path that has more houses than agents or an odd-length alternating path whose end edges are in $C(M')$ because there then exists an agent $a_i$ in $C$ who becomes assigned in $M'$ to a house $h_j$ which is undersubscribed in $M$. Now, since there are no ties in preference lists, $a_i$ must prefer $h_j$ to $M(a_i)$ for otherwise $M' \not\prec M$. However, $M$ is then not trade-in-free, a contradiction. Hence, $C$ must be a cycle. Here, each agent $a_i$ in $C$ is assigned in both $M$ and $M'$ and since $M' \prec M$, each $a_i$ prefers $M'$ to $M$. However, $C$ is then a cyclic coalition with respect to $M$, a contradiction.

It follows that $M' \not\prec M$ and $M$ is Pareto optimal. □

Henceforth we will establish the Pareto optimality of a given matching $M$ in an instance $I$ of CHA by showing that $M$ is maximal, trade-in-free and cyclic-coalition-free. We now show that Lemma 2.3.1 leads to an $O(m)$ algorithm for testing $M$ for Pareto optimality. Let $G$ be the underlying graph of $I$. Then, we can check if $M$ is maximal and trade-in-free in $O(m)$ time by a traversal of the edges in $G$. To check if $M$ is cyclic-coalition-free, we construct the *envy graph* [3] of $M$ as follows. We form a directed graph $G_M$ of $M$ by

**Algorithm 2** Algorithm Greedy-PaCHA

```
1: M := ∅;
2: for each agent a_i in turn do
3:     if there exists some undersubscribed house in a_i's preference list then
4:         let h_j be the most-preferred such house;
5:         M := M ∪ {(a_i, h_j)};
```

letting $G_M$ consist of one vertex for each agent assigned in $M$. We then construct an edge from an agent $a_i$ to another agent $a_j$ in $G_M$ if $a_i$ prefers $M(a_j)$ to $M(a_i)$. It follows that $M$ is cyclic-coalition-free if and only if $G_M$ is acyclic. Note that even though $M$ is a matching of a CHA instance, all vertices in $G_M$ have only unitary capacity (being agent vertices). It follows that a depth-first search suffices to detect any cycles in $O(m)$ time so that these observations lead us to the following lemma.

**Lemma 2.3.2.** *Let $M$ be a matching in a given instance of* CHA*. Then we may check whether $M$ is Pareto optimal in $O(m)$ time.*

Now, given an instance $I$ of CHA, a greedy approach using the serial dictatorship mechanism of [1] gives us a straightforward algorithm, Algorithm Greedy-PaCHA as shown in Algorithm 2, for finding a Pareto optimal matching $M$ in $I$. Here, we consider each agent $a_i$ in turn and give $a_i$ his most preferred house that is currently undersubscribed in the matching built so far. The following lemma shows that the matching constructed by the algorithm must be Pareto optimal.

**Lemma 2.3.3.** *Let $M$ be the matching returned by an execution of Algorithm Greedy-PaCHA. Then, $M$ is Pareto optimal.*

*Proof.* For, suppose not. For each $a_i \in A$, let $A_i$ denote the set of acceptable houses for $a_i$. Consider an agent $a_i$ who is unassigned in $M$. It follows that $A_i$ contains no undersubscribed house $h_j$, otherwise $(a_i, h_j)$ would have been added to $M$, a contradiction. Hence, $M$ is maximal. If $M$ is not trade-in-free, then there exists an agent $a_i$ who prefers some undersubscribed house $h_j$ to $M(a_i)$. This is a contradiction, since $h_j$ must be full at the point when we assign $a_i$ to $M(a_i)$. If $M$ is not cyclic-coalition-free, let us then consider the coalition $C = \langle a_0, a_1, ..., a_{r-1} \rangle$ which exists with respect to $M$. It follows that there exists some agent $a_i$ $(0 \leq i \leq r-1)$ in $C$ who was considered first by the algorithm. By definition of $C$, $a_i$ prefers $M(a_{i+1})$ to $M(a_i)$. Now, $a_{i+1}$ must be considered by the algorithm after $a_i$. However, it follows that $M(a_{i+1})$ must then have had at least one place

| Agent | Pref list | House | Capacity |
|---|---|---|---|
| $a_1$: | $h_1$ $h_2$ | $h_1$ : | 1 |
| $a_2$: | $h_1$ | $h_2$ : | 1 |

Figure 2.1: An instance $I_1$ of CHA

free when $a_i$ was assigned to $M(a_i)$, a contradiction to the fact that the algorithm gives each agent his most preferred undersubscribed house. Hence, $M$ is Pareto optimal. □

The main drawback of Algorithm Greedy-PaCHA is that a given CHA instance may admit Pareto optimal matchings of different cardinalities but Algorithm Greedy-PaCHA may fail to find a Pareto optimal matching of maximum cardinality. For example, Figure 2.1 shows a given CHA instance in which Algorithm Greedy-PaCHA returns a Pareto optimal matching $M_1 = \{(a_1, h_1)\}$ of cardinality 1, given the agent ordering $\langle a_1, a_2 \rangle$, and constructs the maximum Pareto optimal matching $M_2 = \{(a_1, h_2), (a_2, h_1)\}$ of cardinality 2 given the agent ordering $\langle a_2, a_1 \rangle$. It follows that the order in which the agents are considered can have a consequence on the cardinality of the outcome. This is significant from a practical point of view, given that a prime objective in many matching applications is to assign as many agents as possible.

We remark that a straightforward way to find a maximum Pareto optimal matching given a CHA instance $I$ is by constructing a maximum cardinality minimum weight matching as follows. For each edge $(a_i, h_j)$ in the underlying graph $G$ of $I$, we assign a weight $wt(a_i, h_j)$ to the edge by letting $wt(a_i, h_j) = rank_{a_i}(h_j)$ where $rank_{a_i}(h_j)$ denotes the rank of $h_j$ in $a_i$'s preference list. Call this weighted graph $G'$. We then construct a maximum cardinality minimum weight matching in $G'$. The following lemma shows that such a matching must be a maximum Pareto optimal matching in $I$.

**Lemma 2.3.4.** *Let $M$ be a maximum cardinality minimum weight matching in $G'$. Then, $M$ is a maximum Pareto optimal matching in $I$.*

*Proof.* Suppose not. Since $M$ is a maximum matching, it follows that $M$ is maximal. Now, if $M$ is not trade-in-free, then there exists a (agent,house) pair $(a_i, h_j)$ such that $a_i$ is assigned in $M$, $h_j$ is undersubscribed in $M$ and $a_i$ prefers $h_j$ to $M(a_i)$. Consider the matching $M' = (M \backslash (a_i, M(a_i))) \cup (a_i, h_j)$. It is clear that $|M'| = |M|$ and so $M'$ is another maximum cardinality matching of $G'$. However, since $a_i$ prefers $h_j$ to $M(a_i)$, the weight of $M'$ must be smaller than the weight of $M$, a contradiction.

Hence, suppose that $M$ admits some cyclic coalition $C = \langle a_0, a_1, ..., a_{r-1} \rangle$. Let $M'$ be the matching obtained by satisfying $C$. Then, it is clear that $|M'| = |M|$ again. Moreover, since each $a_i$ prefers $M(a_i)$ to $M'(a_i)$ for $0 \leq i \leq r-1$, the weight of $M'$ is again smaller than the weight of $M$, a contradiction. □

Note that the above lemma also indicates that a maximum Pareto optimal matching in $I$ has the same cardinality as a maximum matching in $G$ and any maximum cardinality minimum weight matching of $G'$ gives us a maximum Pareto optimal matching in $I$.

A well known transformation in matching theory (described in [42]) allows us to transform the problem of finding a maximum cardinality minimum weight matching into the Assignment problem. Recall from Section 1.2 that we can solve the Assignment problem in the capacitated bipartite graph in $O(C \min(m \log n, n^2))$ time [15], so this allows us to find a maximum Pareto optimal matching in the same time complexity. However, since the problem of finding a maximum matching in the capacitated bipartite graph takes $O(\sqrt{C}m)$ time (as mentioned in Section 1.2), it is of interest to consider whether faster algorithms for finding a maximum Pareto optimal matching in CHA exist.

## 2.4 Maximum Pareto optimal matchings

In this section, we describe a three-phase algorithm for finding a maximum Pareto optimal matching in CHA by satisfying the necessary and sufficient conditions in Lemma 2.3.1. Let $I$ be an instance of CHA and $G$ be its underlying graph. The problem of finding a maximum matching in $G$ can be viewed as an instance of maximum cardinality DCS [15] as described in Section 1.2 (the two problems are essentially the same, except that agents have no explicit preferences in the DCS case; the definition of a matching is unchanged). Hence, Phase 1 of the algorithm uses Gabow's algorithm [15] to compute a maximum matching $M$ in $G$. This phase guarantees that $M$ is maximal and takes $O(\sqrt{C}m)$ time. The next two phases ensure that $M$ is trade-in-free and cyclic-coalition-free respectively as detailed below.

### 2.4.1 Phase 2 of the algorithm

In this phase, we transform $M$ into a trade-in-free matching by conducting a repeated search for (agent,house) pairs $(a_i, h_j)$ such that $h_j$ is undersubscribed in $M$ and $a_i$ prefers $h_j$ to $M(a_i)$. Whenever such a pair is found, the algorithm breaks the existing assignment

**Algorithm 3** Phase 2 loop

```
1: while S ≠ ∅ do
2:     h_j := S.pop();
3:     (a_i, r) := L_j.removeHead();
4:     if r < curr_{a_i} then
5:         h'_j := M(a_i);
6:         M := (M \ {(a_i, h'_j)}) ∪ {(a_i, h_j)};
7:         curr_{a_i} := r;
8:         if |M(h_j)| < c_j and L_j ≠ ∅ then
9:             S.push(h_j);
10:        h_j := h'_j;
11:    if L_j ≠ ∅ and h_j ∉ S then
12:        S.push(h_j);
```

between $a_i$ and $M(a_i)$, and promotes $a_i$ to $h_j$. It follows that a space in $M(a_i)$ becomes freed in the process, which may consequently be assigned to some assigned agent $a_k$ who prefers $M(a_i)$ to $M(a_k)$. Note that if $h_j$ remains undersubscribed after such a step, it may also be assigned to some assigned agent $a_l$ who prefers $h_j$ to $M(a_l)$. We show how to obtain a trade-in-free matching from $M$ by using a slight modification of the Phase 2 loop of the algorithm described by Abraham et al. [3] to find a maximum Pareto optimal matching in HA.

For each house $h_j$, we maintain a linked list $L_j$ of pairs $(a_i, r)$ where $a_i$ is an assigned agent who prefers to be assigned to $h_j$ than $M(a_i)$ at the start of Phase 2, and $r$ is the rank of $h_j$ in $a_i$'s preference list. Note that the pairs in $L_j$ may subsequently contain an agent $a_i$ who prefers $M(a_i)$ to $h_j$ if $M(a_i)$ is no longer the house that $a_i$ was assigned to at the start of Phase 2 as a result of promotions executed over the course of the algorithm. We will maintain a stack $S$ of all undersubscribed houses $h_j$ where $L_j$ is non-empty. Also, for each house $h_j$, we assume that we store a counter for $|M(h_j)|$. For each assigned agent $a_i$, let $curr_{a_i}$ be a variable which stores the rank of $M(a_i)$ in $a_i$'s preference list.

Let us now consider the pseudocode of the Phase 2 loop as shown in Algorithm 3. During each iteration of the main while loop, we pop an undersubscribed house $h_j$ from $S$ and remove the first pair $(a_i, r)$ from $L_j$ (which must be non-empty). Now, if $r < curr_{a_i}$, it follows that $a_i$ prefers $h_j$ to $M(a_i)$ so we promote $a_i$ from $h'_j = M(a_i)$ to $h_j$ and we update $M$ and $curr_{a_i}$ in the process. Now, if $h_j$ remains undersubscribed at the end of this step, then we push $h_j$ back onto $S$ if $L_j$ is non-empty. We also push $h'_j$ onto $S$ if $L_{h'_j}$

is non-empty and if $h'_j$ is not already in $S$. Otherwise, if $r \geq curr_{a_i}$, we push $h_j$ back onto $S$ if $L_j$ is non-empty.

Now, the algorithm must terminate, for each iteration of the main while loop removes a pair from a list $L_j$ but no new pair is ever added to any list during a loop iteration. Hence, the algorithm terminates when $S$ is empty. It must be the case that when this happens no assigned agent $a_i$ prefers an undersubscribed house to $M(a_i)$, so that $M$ is trade-in-free as a result. Moreover, since each agent assigned at the end of Phase 1 is also assigned at the end of Phase 2, $M$ remains a maximum matching. Let us then consider the time complexity of Phase 2. We can initialise all variables used in the Phase 2 loop in $O(m)$ time using a single traversal of the agents' preference lists. The number of iterations of the main while loop is bounded above by the total length of preference lists. It is straightforward to verify that each operation within the while loop takes constant time (with a suitable choice of data structures such as those described later in Section 2.4.3). Hence, the algorithm runs in $O(m)$ time, giving us the following result.

**Lemma 2.4.1.** *Given a maximum matching $M$ in an instance of* CHA*, the Phase 2 loop ensures that $M$ is trade-in-free in $O(m)$ time.*

### 2.4.2 Phase 3 of the algorithm

In this phase, we transform $M$ into a matching $M'$ that admits no cyclic coalition by using a modification of the linear-time extension [3] of Gale's Top Trading Cycles Method [57]. This phase consists of a pre-processing step which we will describe in detail, and then the main Phase 3 loop shown in Algorithm 5. Throughout Phase 3, we maintain a stack of agents $P$ which will help us to identify cyclic coalitions. The matching $M'$ and the stack $P$ are empty at the start of Phase 3. For each agent $a_i$, we maintain a pointer $p(a_i)$ to the first house on $a_i$'s preference list, and subsequently $p(a_i)$ traverses left to right over the course of execution of Phase 3. We will also maintain a queue of agents $Q$, each of whom is an agent $a_i$ waiting to be assigned to $p(a_i)$ in $M'$. In addition, for each house $h_j$, we will use $M_0(h_j)$ to store those agents who are assigned to $h_j$ in $M$ but who are unassigned in $M'$ so far in the execution of Phase 3. Initially, $M_0(h_j)$ will contain all those agents assigned to $h_j$ in $M$. As we assign agents in $M_0(h_j)$ to houses in $M'$, we will remove these agents from $M_0(h_j)$. Finally, we also maintain a linked list $L_j$ for each house $h_j$ containing agents such that if $a_i$ is an agent in $L_j$, then $a_i$ prefers $h_j$ to $M(a_i)$.

**Algorithm 4** Process (Q)

```
1: while Q ≠ ∅ do
2:    a_i := Q.removeHead();
3:    h_j := p(a_i);
4:    h_k := M(a_i); {// possibly h_j = h_k}
5:    M' := M' ∪ {(a_i, h_j)};
6:    label a_i;
7:    if a_i ∈ P then
8:       remove a_i from P;
9:    M_0(h_k) := M_0(h_k) \ {a_i};
10:   if |M'(h_j)| = c_j then
11:      for each unlabelled a'_i ∈ L_j do
12:         delete h_j from the preference list of a'_i;
13:         if p(a'_i) = M(a'_i) then
14:            Q.add(a'_i);
```

#### 2.4.2.1 Pre-processing step

Let us now introduce the pre-processing step which helps to reduce the number of iterations of the Top Trading Cycles Method in the main Phase 3 loop. This step makes use of the observation (as in [3]) that no agent $a_i$ assigned to his first choice house $h_j$ in $M$ can be involved in a cyclic coalition. At the outset of Phase 3, we check if $p(a_i) = M(a_i)$ for each agent $a_i$ and add every such $a_i$ to $Q$.

If $Q$ is non-empty, then we run the sub-routine Process(Q), shown in Algorithm 4, as the pre-processing step. Note that this usage of Process(Q) is prior to the main Phase 3 loop starting, but it will be used again in general during the main Phase 3 loop. This sub-routine considers each agent $a_i$ in $Q$ in turn, by removing $a_i$ from $Q$ and then adding the edge $(a_i, h_j)$ to $M'$. Every such $a_i$ is then labelled to differentiate $a_i$ from those agents unassigned in $M'$ so far in the execution of the algorithm (all agents are initially unlabelled at the outset of Phase 3). Now, $P$ must be empty during pre-processing. However, this may not be true during a subsequent execution of Process(Q) by the main Phase 3 loop. Hence, Process(Q) checks if $a_i$ lies in $P$, and if so, removes $a_i$ from $P$ so as to remove the agent from further consideration by the main Phase 3 loop, since $a_i$ has just been assigned in $M'$. Let $p(a_i) = h_j$. Now, if $|M'(h_j)| = c_j$ after the assignment of $a_i$ to $h_j$, then we remove $h_j$ from the preference lists of the remaining agents since such a house that is full in $M'$ could not subsequently be involved in a cyclic coalition. We refer to those preference

lists in which houses have been removed as *reduced preference lists*. We then apply the observation made at the start of this subsection recursively to the reduced preference lists of the remaining agents until either (i) no agents remain unassigned in $M'$, or (ii) at least one agent is not assigned to his reduced first choice in $M'$ by Process(Q). In case (i), each agent is assigned to his reduced first choice (i.e. the first choice on his reduced preference list) in $M'$ and so cannot be involved in any cyclic coalition as Lemma 2.4.4 on page 31 will establish. The following lemma shows that when case (ii) happens at the end of the pre-processing step, a cyclic coalition must exist with respect to $M$.

**Lemma 2.4.2.** *Suppose that pre-processing terminates, and there exists an agent*[1] *who is unassigned in $M'$. Then a cyclic coalition must exist with respect to $M$.*

*Proof.* Let $a_0$ be an agent who is not assigned in $M'$ to his reduced first choice $p(a_0)$ at the end of pre-processing. Hence, $a_0$ is an unlabelled agent and $p(a_0) \neq M(a_0)$. It follows that $p(a_0)$ must be full in $M$ for otherwise $M$ is not trade-in-free, a contradiction. However, $p(a_0)$ cannot be full in $M'$ for otherwise $p(a_0)$ would have been removed from $a_0$'s preference list by pre-processing and cannot be the reduced first-choice house of $a_0$. Hence, there exists some agent $a_1 \in M(p(a_0))\backslash M'(p(a_0))$ because if an agent $a$ is assigned in $M'$ by Process(Q) in pre-processing, it must be the case that $a$ must be assigned in $M'$ to $M(a)$. It follows immediately that $a_1$ must be unassigned in $M'$. Furthermore, $p(a_1) \neq M(a_1)$ (or else $a_1 \notin M(p(a_0))\backslash M'(p(a_0))$) so that $p(a_0) \neq p(a_1)$. By reusing the same argument, it follows that we can trace a sequence of agents $S = \langle a_0, a_1, ...\rangle$ such that $a_i$ is assigned in $M$ but unassigned in $M'$ and $p(a_i) = M(a_{i+1})$ for $i \geq 0$. Since the number of agents is finite, there must be some $r$ such that $a_r = a_x$ for some $0 \leq x < r-1$, where without loss of generality $a_x, a_{x+1}, ..., a_{r-1}$ are distinct agents. However, the substring of agents $C = \langle a_x, a_{x+1}, ..., a_{r-1}\rangle$ within $S$ must then constitute a cyclic coalition with respect to $M$. □

Now, it is clear that an (unlabelled) agent $a_i$ can only be added to $Q$ when the last house that $a_i$ prefers to $M(a_i)$ gets removed from his preference list so that $p(a_i)$ becomes equal to $M(a_i)$, and this happens only once in pre-processing. Since no agent is added to $Q$ twice, the while loop of process(Q) is bound to terminate. As a result, the pre-processing step must also terminate.

[1] in fact, if there exists one such agent, then this lemma proves that there will be at least two such agents

**Algorithm 5** Main Phase 3 loop

```
1: for each unlabelled agent a_i do
2:    P := {a_i}; {// P is a stack of agents}
3:    c(a_i) := 1; {// counter record the number of times an agent is in P}
4:    while P ≠ ∅ do
5:       a'_i := P.pop();
6:       if c(a'_i) = 2 then
7:          a''_i := a'_i;
8:          repeat
9:             Q.add(a''_i);
10:            a''_i := P.pop();
11:         until a''_i = a'_i
12:         call Process(Q);
13:      else
14:         P.push(a'_i);
15:         choose any a''_i ∈ M_0(p(a'_i));
16:         c(a''_i) := c(a''_i) + 1;
17:         P.push(a''_i);
```

#### 2.4.2.2 Phase 3 loop

We then make use of the algorithm in the main Phase 3 loop, as shown in Algorithm 5 to construct the envy graph in order to detect and satisfy cyclic coalitions. For each agent $a_i$ who is not assigned to his reduced first-choice in $M$, we repeatedly build a path of agents (represented by $P$) starting from $a_i$ in the main while loop and check if $P$ cycles. To do so, we initialise a counter $c(a_i)$ to 0 for each agent $a_i$.

Now, if $c(a'_i) \neq 2$ for some agent $a'_i$ in $P$ during an iteration of the while loop, then we extend $P$ by following the reduced first-choice edge of $a'_i$ in line 15. Let $p(a'_i) = h_j$ and let $a''_i$ be any member of $M_0(h_j)$. Note that $M_0(h_j)$ must be non-empty. For, suppose not. Since $a'_i$ prefers $h_j$ to $M(a'_i)$, $h_j$ must be full in $M$ (or else $M$ is not trade-in-free). Each agent $a_k$ assigned to $h_j$ in $M$ either becomes assigned to $h_j$ again in $M'$ if $p(a_k) = M(a_k)$, or to some other house via the satisfaction of some cyclic coalition. In the latter case, this causes some agent $a_l$ to be assigned to $h_j$ in $M'$ in $a_k$'s place. Since $h_j$ is not full in $M'$ by definition of $p(a'_i) = h_j$, it follows that there exists some agent belonging to $M(h_j)$ who is currently unassigned in $M'$. Hence, $M_0(h_j)$ must be non-empty.

Otherwise if $c(a'_i) = 2$, it follows that we have a cyclic coalition in $P$ starting from $a'_i$.

We satisfy $C$ by popping each agent $a_i''$ in $C$ from $P$ until we remove $C$, and add each $a_i''$ to $Q$. We then call Process(Q) to assign each $a_i''$ to $p(a_i'')$ in $M'$, to label each $a_i''$ in order to remove the agent from further consideration by the algorithm, as well as to remove $M'(a_i'')$ from the preference lists of the remaining unlabelled agents if the house becomes full in $M'$.

#### 2.4.2.3 Correctness of Phase 3 loop

If there are unlabelled agents at the start of the main Phase 3 loop, there must exist at least one cyclic coalition with respect to $M$ involving a subset of these agents by Lemma 2.4.2. The following lemma strengthens this result by showing that if there exist any unlabelled agents at any point of time in the execution of Phase 3, then a cyclic coalition must exist.

**Lemma 2.4.3.** *Consider a given iteration of the for loop in Phase 3. If there exists an agent who remains unlabelled, then a cyclic coalition must exist with respect to $M$.*

*Proof.* Let $a_0$ be an agent who is unlabelled during a given iteration of the for loop of Phase 3. Then, $a_0$ is not assigned in $M'$ to his reduced first choice $p(a_0)$. It follows that $p(a_0) \neq M(a_0)$. Now, $p(a_0)$ must be full in $M$ for otherwise $M$ is not trade-in-free, a contradiction. However, $p(a_0)$ cannot be full in $M'$ for otherwise $p(a_0)$ would have been removed from $a_0$'s preference list by Process(Q) and cannot be the reduced first-choice house of $a_0$. Now, each agent $a' \in A$ becomes assigned in $M'$ to either $M(a')$ when $M(a') = p(a')$, or to $p(a')$ when we satisfy a cyclic coalition involving $a'$. Since $p(a_0)$ is currently undersubscribed in $M'$, it follows that there exists a non-empty subset of agents $A_s$ such that each agent $a$ in $A_s$ belongs to $M(p(a_0))\backslash M'(p(a_0))$ and $a$ is currently unassigned in $M'$. Let $a_1 \in A_s$. It must be the case that $p(a_1) \neq M(a_1)$ and hence, $p(a_1) \neq p(a_0)$. By reusing the same argument, it follows that we can trace a sequence of agents $S = \langle a_0, a_1, ... \rangle$ such that $a_i$ is assigned in $M$ but unassigned in $M'$ and $p(a_i) = M(a_{i+1})$ for $i \geq 0$. Since the number of agents is finite, there must be some $r$ such that $a_x = a_r$ for some $0 \leq x < r - 1$, where without loss of generality $a_x, a_{x+1}, ..., a_{r-1}$ are distinct agents. However, the substring of agents $C = \langle a_x, a_{x+1}, ..., a_{r-1} \rangle$ within $S$ must then constitute a cyclic coalition with respect to $M$. □

The next lemma shows that when all agents are assigned in $M'$, we then obtain a cyclic-coalition-free matching.

**Lemma 2.4.4.** *If no unlabelled agents remain at any stage of Phase 3, then $M'$ is cyclic-coalition-free.*

*Proof.* If there does not exist any unlabelled agents, then every agent is assigned in $M'$. Let $a_0$ be an arbitrary agent. Suppose that there is a cyclic coalition $C = \langle a_0, a_1, ..., a_{r-1} \rangle$ with respect to $M'$ involving $a_0$. Let $M'(a_0) = h_j$ and let $M'(a_1) = h_k$. By definition of $C$, $a_0$ must prefer $h_k$ to $h_j$. Since $a_0$ was assigned to $h_j$ instead of $h_k$ in $M'$, it follows that $h_k$ must have been full in $M'$ at the time that $a_0$ was assigned to $h_j$ in $M'$. It must then be the case that $a_1$ was considered by Phase 3 before $a_0$ or else $M'(a_1) \neq h_k$. Now, by applying the same argument to the remaining agents in $C$, we can establish that $a_{r-1}$ must have been considered by Phase 3 before $a_0$. Let $M'(a_{r-1}) = h_l$. It follows that $a_{r-1}$ must prefer $h_j$ to $h_l$. Now, it must be the case that at the time that $a_{r-1}$ was assigned in $M'$, $h_j$ must have been undersubscribed for otherwise $a_0$ could not have been assigned to $h_j$ later on. However, this gives a contradiction for $a_{r-1}$ prefers $h_j$ to $h_l$ and should then be assigned to $h_j$ by Phase 3 instead. □

Suppose that the envy graph involves the sequence of agents $S = \langle a_0, a_1, ..., a_{r-1} \rangle$ and suppose that only a substring of these agents $C = \langle a_i, a_{i+1}, ..., a_{r-1} \rangle$ constitute a cyclic coalition where $0 \leq i < r - 1$. Let us call the agents in the substring $\langle a_0, a_1, ..., a_{i-1} \rangle$ the *tail* of $C$. Now, if certain houses become full in $M'$ as a result of satisfying $C$, thereby causing $M(a_{i-1})$ to become the reduced first choice house for the agent $a_{i-1}$, then $a_{i-1}$ gets added to $Q$ and assigned to $M(a_{i-1})$ in $M'$ subsequently by Process(Q). Note that this can cause an unwinding effect in the tail in which each agent $a_k$ $(0 \leq k \leq i - 2)$, such that $M(a_k)$ lies immediately after $p(a_k)$ in $a_k$'s reduced preference list, gets added to $Q$ and assigned to $M(a_k)$ in $M'$ by Process(Q) in descending agent subscript order until either we reach an agent $a_{k-1}$ such that there exists a house between $p(a_{k-1})$ and $M(a_{k-1})$ in $a_{k-1}$'s reduced preference list or the tail becomes empty as a result. In the former case, the main Phase 3 loop then extends $P$ by following the reduced first-choice edge of $a_{k-1}$. In the latter case, the main Phase 3 loop tries to extend $P$ by following the reduced first-choice edge of the next unlabelled agent, if one exists.

It is straightforward to see that the labelling of agents and the maintenance of $c(a_i)$ for each agent $a_i$ ensures that no agent $a_i$ is added more than twice to $P$ in Phase 3. Clearly, if $P$ is non-empty, $P$ must cycle at some point of time in the execution of Phase 3 (as observed by Lemma 2.4.3). Since each agent $a_i$ that we add to $P$ belongs to a cyclic

coalition or to the tail of a cyclic coalition, we are bound to remove $a_i$ from $P$ as a result. It also follows by the labelling of agents that any agent added to $Q$ by the for loop in the main Phase 3 loop is not added to $Q$ again by Process(Q) and vice versa. Hence, no agent gets added to $Q$ twice.

Hence, Phase 3 must terminate when no unlabelled agents remain. When this happens, it follows by Lemma 2.4.4 that $M'$ must be a cyclic-coalition-free matching. We next show that each agent $a_i$ assigned in $M$ at the end of Phase 2 must also be assigned in $M'$ at the end of Phase 3.

**Lemma 2.4.5.** *Each agent $a_i$ assigned in $M$ at the end of Phase 2 is also assigned in $M'$ at the end of Phase 3.*

*Proof.* Suppose not. Then, let $a_i$ be an agent who is unassigned in $M'$. Let $M(a_i) = h_j$. Then, $a_i \in M(h_j) \backslash M'(h_j)$. It follows that $h_j$ cannot have been the first house on $a_i$'s preference list, or else pre-processing would have assigned $M'(a_i)$ to be $h_j$. Hence, there exists at least one house that $a_i$ prefers to $h_j$. Now, if $a_i$ is not assigned in $M'$ to any of these houses, then the pointer $p(a_i)$ should move across $a_i$'s preference list until it points at $h_j$. When this happens, $a_i$ should then be assigned to $h_j$ in $M'$. However, $a_i$ is unassigned in $M'$ so that $h_j$ must have been removed from $a_i$'s preference list prior to this as a result of it becoming full in $M'$. Now, if $h_j$ is full in $M'$, then for every $a_k \in M'(h_j)$, either $a_k \in M(h_j)$ or there exists a unique $a_l \in M(h_j) \backslash M'(h_j)$ such that $a_k$ and $a_l$ belong to the same cyclic coalition. However, this implies that $c_j + 1$ agents were assigned to $h_j$ in $M$, a contradiction. □

Since $M$ is a matching that is also maximum, it follows by Lemma 2.4.5 that $M'$ is also a maximum matching.

### 2.4.3 Implementation and analysis

The time complexity of Phase 3 depends on how efficiently we can implement Process(Q) and the main Phase 3 loop. Let us consider briefly the data structures required.

First of all, let us assume that we represent the stack $P$ as a doubly linked list emdedded within an array. We let $P$ contain $n_1$ elements and we indicate the presence or absence of an agent in $P$ by a 1 or 0 respectively. We maintain a pointer to the top of the stack in addition to previous and next pointers between agents in $P$ at any point of time. We implement $Q$ as a straightforward linked list. We also represent the preference list $pref_{a_i}$

of each agent $a_i$ as a doubly linked list embedded within an array. Each element in position $j$ of $pref_{a_i}$ stores the house lying in the corresponding position in $a_i$'s preference list. Also, for each house $h$ in $pref_{a_i}$, we maintain a pointer each to the house preceding and following $h$ on $a_i$'s preference list respectively. We let $p(a_i)$ point to the first house in $pref_{a_i}$, i.e. $a_i$'s (reduced) first choice house. Note that as with any doubly linked list, $p(a_i)$ gets updated as part of the deletion operation whenever the (reduced) first choice house in $pref_{a_i}$ is to be deleted from $pref_{a_i}$. We then build a rank array $rank_{a_i}$ for each agent $a_i$ which stores the position of each house $h_j$ on $a_i$'s preference list, i.e. $rank_{a_i}[j]$ gives the position of $h_j$ on $pref_{a_i}$.

For each house $h_j$, we represent $M_0(h_j)$ also as a doubly linked list embedded within an array. There are $n_1$ entries in each $M_0(h_j)$ indexed according to agent subscript, but we maintain previous and next pointers for at most $c_j$ of these entries, i.e. previous and next pointers exist between consecutive agents in $M_0(h_j)$ only if these agents belong to $M(h_j)$ but are currently unassigned in $M'$. It is straightforward to see that $L_j$ for each $h_j$ can be implemented as a linked list. We also maintain a counter $|M'(h_j)|$ for each house $h_j$ to keep track of the number of agents assigned to it in $M'$ so far in the execution of Phase 3.

Note that for each doubly linked list that we use, we let the previous pointer of the first element and the next pointer of the last element to each point to null. Now, if we use virtual initialisation (described in [9, p.149]) for the initialisation of $P$, it is clear that it takes only a single traversal of the agents' preference lists in agent subscript order to initialise the rest of the data structures.

To illustrate the use of these data structures, suppose firstly that we pop an agent $a_i$ from $Q$. It is straightforward to see if $a_i$ belongs to $P$ by checking if $P[i]$ is 1 or 0. To remove $a_i$ from $P$, we set $P[i] = 0$ and update the previous and next pointers of $P[i]$'s predecessor and successor, as well as the pointer to the top of $P$ if necessary. Let $h_k = M(a_i)$. To remove $a_i$ from $M_0(h_k)$, we update the next and previous pointers of $M_0(h_k)[i]$'s predecessor and successor respectively. Let $p(a_i) = h_j$. If $h_j$ becomes full in $M'$ as a result of assigning $a_i$ to $h_j$, i.e. $|M'(h_j)| = c_j$, we want to remove $h_j$ from the preference lists of those unlabelled agents in $L_j$. It is clear that it takes only time linear in the size of $L_j$ to find these agents. Let $a_l$ be such an agent. Then, $rank_{a_l}[j]$ enables us to look up the position of $h_j$ on $a_l$'s preference list in constant time. We then delete $h_j$ from $pref_{a_l}$ by updating the pointers of $h_j$'s predecessor and successor in $pref_{a_l}$. In

this way, $p(a_l)$ always points to the head of $a_l$'s preference list (which must be currently undersubscribed in $M'$).

Let us then consider the time complexity of Phase 3 by looking at the entire execution of the main Phase 3 loop and Process(Q) taken over the algorithm's execution. It has already been observed that each agent can be added to $P$ no more than twice and added to $Q$ at most once. Now, lines 8-11 of the main Phase 3 loop are executed at most once for each agent in $P$. All other operations in the main Phase 3 loop are just $O(1)$ stack operations or simple manipulation of data structures. In Process(Q), all operations apart from those in lines 11-14 can be implemented to run in $O(1)$ time for each agent that is added to or removed from $Q$. Finally, it is clear that lines 11-14 are executed at most once for each house, and hence, the total number of iterations of the for loop is $O(m)$, taken over all calls to Process(Q). It follows that Phase 3 takes $O(m)$ time overall, giving us the following result.

**Lemma 2.4.6.** *Given a maximal trade-in-free matching $M$ in an instance of* CHA*, Phase 3 constructs a cyclic-coalition-free matching $M'$ from $M$ in $O(m)$ time.*

Since Phase 1 dominates the overall complexity of the algorithm to find a maximum Pareto optimal matching in an instance of CHA, we have the following result.

**Theorem 2.4.1.** *Given an instance $I$ of* CHA*, we can find a maximum Pareto optimal matching in $I$ in $O(\sqrt{C}m)$ time.*

Recall that an alternative approach to finding a Pareto optimal matching given an CHA instance $I$ involves obtaining a weighted graph $G'$ of $I$ and then finding a maximum cardinality minimum weight matching of $I$. This takes $\Omega(C \min(m \log n, n^2))$ time. If $m \log n \leq n^2$, then it follows that our algorithm is faster by a factor of $\Omega(\sqrt{C} \log n)$. Otherwise, $m \log n > n^2$ and our algorithm is faster by a factor of $\Omega(\sqrt{C}n^2/m)$.

## 2.5 Open problem

We conclude with the following open problem.

The problem model of CHA defined in Section 2.2 can be generalized by permitting agents to contain ties in their preference lists, i.e. CHAT. In this context, the definition of the relation $\prec$ is the same as that given in Section 1.3.2.1, and hence the definition of Pareto optimality remains unchanged. A maximum Pareto optimal matching can be

found in $O(C \min(m \log n, n^2))$ time [15] by transformation to the Assignment problem as described in Section 2.3. However, is the problem of finding a maximum Pareto optimal matching also solvable in $O(\sqrt{C}m)$ time in the presence of ties?

# Chapter 3

# Pareto optimal matchings in HR

## 3.1 Introduction

In this chapter, we extend our study of Pareto optimal matchings from Chapter 2 to the bipartite model with two-sided preferences. We focus our attention on the Hospitals-Residents problem without ties (HR), which was introduced in Section 1.4.2.

The main results of this chapter, and their organisation are as follows. We give some terminology and preliminary results on Pareto optimal matchings in HR in Section 3.2. We then give a characterisation of Pareto optimal matchings in HR in Section 3.3, which we subsequently use in Section 3.4 to construct an $O(\sqrt{C}m)$ time algorithm for finding a maximum Pareto optimal matching given an instance $I$ of HR, where $C$ is the total capacity of the hospitals and $m$ is the total length of preference lists in $I$. Finally, in Section 3.5, we show how to adapt our algorithm for HR to obtain a faster $O(\sqrt{n}m)$ time algorithm to solve the analogous problem given an instance of SMI, a special case of HR, where $n$ is the total number of men and women.

## 3.2 Basic terminology and preliminary results

Let $I$ be an instance of HR. We reuse the notations and terminology for HR as defined in Section 1.4.2, and also provide some additional definitions as follows.

Given a resident $r_i \in R$ and an acceptable hospital $h_j$ for $r_i$, we define $rank_{r_i}(h_j)$ to be the number of hospitals that $r_i$ prefers to $h_j$ plus 1. If $rank_{r_i}(h_j) = k$, we say that $h_j$ is the *kth choice* of $r_i$. In a similar way, we define $rank_{h_j}(r_i)$ and the $k$th choice of $h_j$. We assume that no resident has an empty preference list and each hospital is acceptable

to at least one resident so that $m \geq \max\{n_1, n_2\}$. Also, let $M$ be a matching in $I$. We define a vertex $v$ in $G$ to be *exposed* with respect to $M$ if either (i) $v$ is a resident vertex that is unassigned in $M$, or (ii) $v$ is a hospital vertex that is undersubscribed in $M$. An *augmenting path* in $G$ is an alternating path, both of whose end vertices are exposed.

Let $M'$ be another matching in $I$. Recall from Section 1.4.2 that a resident $r_i$ *prefers* $M$ to $M'$ if either (i) $r_i$ is assigned in $M$ and unassigned in $M'$, or (ii) $r_i$ is assigned in both $M$ and $M'$ and prefers $M(r_i)$ to $M'(r_i)$. Furthermore, a hospital $h_j$ *prefers* $M$ to $M'$ if

- $|M(h_j)| > |M'(h_j)|$, or
- $|M(h_j)| = |M'(h_j)|$ and $h_j$ prefers the worst resident assigned to it in $M$ to the worst resident assigned to it in $M'$.

Unlike the case for residents where it is necessary for ties to be present in the preference lists in order for a resident to be indifferent between any two matchings $M$ and $M'$, a hospital $h_j$ may be indifferent between $M$ and $M'$ if $|M(h_j)| = |M'(h_j)|$, the worst resident assigned to $h_j$ is the same in both $M$ and $M'$ but $h_j$ has different sets of residents assigned to it in $M$ and $M'$.

Given these definitions, we may define a relation $\prec$ on the set of all matchings in $I$ as in Section 1.3.2.1: that is, $M \prec M'$ if and only if no agent prefers $M'$ to $M$, and some agent prefers $M$ to $M'$. A matching $M$ is defined to be *Pareto optimal* if and only if it is $\prec$-minimal. In other words, a matching $M$ is Pareto optimal if there is no other matching $M'$ such that $M' \prec M$.

## 3.3 Characterisation of Pareto optimal matchings in HR

Let $M$ be a matching in $I$. The following defines a necessary and sufficient condition for $M$ to be Pareto optimal in $I$.

**Definition 3.3.1.** *An* improving coalition *with respect to $M$ is a sequence of agents $C = \langle r_0, h_0, r_1, h_1, ..., r_{k-1}, h_{k-1}\rangle$, for some $k \geq 1$, such that the residents are distinct and:*

1. $(r_i, h_{i-1}) \in M$ $(1 \leq i \leq k-1)$,
2. *Either*

(a) $r_0$ *is unassigned in* $M$ *and finds* $h_0$ *acceptable, and* $h_{k-1}$ *is undersubscribed in* $M$ *and finds* $r_{k-1}$ *acceptable, or*

(b) $k \geq 2$, *and* $(r_0, h_{k-1}) \in M$.

3. $r_i$ *prefers* $h_i$ *to* $h_{i-1}$ *for each* $i$ $(1 \leq i \leq k-1)$, *and* $r_0$ *prefers* $h_0$ *to* $h_{k-1}$ *if Condition 2(b) holds,*

4. $h_i$ *prefers* $M'$ *to* $M$ *or is indifferent between the matchings for each* $i$ $(0 \leq i \leq k-1)$ *where* $M' = M \oplus C$.

*If* $M$ *admits no improving coalition, we say that* $M$ *is* improving coalition-free.

If $C$ satisfies Condition 2(a), we also refer to $C$ as an *augmenting coalition*, otherwise we also refer to $C$ as a *cyclic coalition*. We define the *size* of $C$ to be $2k$, and henceforth, all subscripts are taken modulo $k$ when reasoning about improving coalitions. The matching $M'$ obtained by $M' = M \oplus C$ is defined to be the matching obtained from $M$ by *satisfying* $C$. Note that the hospitals may be repeated in $C$ in view of their non-unitary capacities. A matching $M$ is *maximal* in $G$ if $M \cup \{e\}$ is not a matching for any $e \in E \backslash M$ where $E$ is the edge set in $I$. By Definition 3.3.1, $M$ is maximal if and only if $M$ admits no improving coalition of size 2. The following lemma gives a necessary and sufficient condition for a matching in HR to be Pareto optimal.

**Lemma 3.3.1.** *Let* $M$ *be a matching in a given instance* $I$ *of HR. Then* $M$ *is Pareto optimal if and only if* $M$ *is improving coalition-free.*

*Proof.* Let $M$ be a Pareto optimal matching. If $M$ admits some improving coalition $C$, let $M'$ be the matching obtained by satisfying $C$. By Definition 3.3.1, each resident in $C$ prefers $M'$ to $M$ and each hospital in $C$ either prefers $M'$ to $M$ or is indifferent between the two matchings. However, this implies that $M' \prec M$, a contradiction.

Conversely, let $M$ be a matching that is improving coalition-free, and suppose for a contradiction that $M$ is not Pareto optimal. Then there exists some matching $M'$ such that $M' \prec M$. Let $X = M \oplus M'$ and let $C$ be a connected component of $X$. We consider three cases according to the form of $C$.

– *Case (i):* $C$ is an alternating path with an even number of edges. However, this implies that either there exists a resident $r_i$ who is assigned in $M$ but who becomes unassigned in $M'$, or there exists a hospital $h_j$ such that $|M'(h_j)| < |M(h_j)|$. Both possibilities contradict the fact that $M' \prec M$.

– *Case (ii):* $C$ is an alternating path with an odd number of edges. Clearly, the end edges of $C$ cannot be in $M$ for otherwise we can obtain a similar contradiction as in Case (i). Hence, the end edges of $C$ are in $M'$. Let $r_i$ be the exposed resident vertex in $C$. Clearly, $r_i$ prefers $M'$ to $M$. For each resident $r_j \neq r_i$ in $C$, it must be the case that $r_j$ prefers $M'(r_j)$ to $M(r_j)$ since $M' \prec M$. Hence, every resident in $C$ prefers $M'$ to $M$. Let $h_x$ be the exposed hospital vertex in $C$. Clearly, $|M'(h_x)| > |M(h_x)|$ so that $h_x$ prefers $M'$ to $M$ by Definition 1.4.1(i). For each hospital $h_y \neq h_x$ in $C$, it is clear that $|M'(h_y)| = |M(h_y)|$. Furthermore, $h_y$ must either prefer $M'$ to $M$ or is indifferent between the matchings according to Definition 1.4.1(ii), or else it cannot be the case that $M' \prec M$. However, $C$ is then an augmenting coalition with respect to $M$.

– *Case (iii):* $C$ is an alternating cycle. Clearly, each resident $r_i$ in $C$ is assigned in both $M$ and $M'$. Since $M' \prec M$, it must be the case that each $r_i$ prefers $M'(r_i)$ to $M(r_i)$. For each hospital $h_j$ in $C$, it is clear that $|M'(h_j)| = |M(h_j)|$. Furthermore, $h_j$ must either prefer $M'$ to $M$ or is indifferent between the matchings according to Definition 1.4.1(ii), or else it cannot be the case that $M' \prec M$. However, $C$ is then a cyclic coalition with respect to $M$, a contradiction. □

Henceforth we will establish the Pareto optimality of a given matching $M$ in an instance $I$ of HR by showing that $M$ is improving coalition-free. We now show that Lemma 3.3.1 leads to an $O(m)$ algorithm for testing a given matching in an HR instance for Pareto optimality. Let $M$ be a matching in an HR instance $I$ and let $G$ be the underlying graph of $I$. We first perform the following transformation to the preference lists of agents. That is, for every resident $r_i \in R$, we remove the hospital $h_j$ from the preference list of $r_i$, and remove $r_i$ from the preference list of $h_j$ if $h_j$ is a hospital that lies after $M(r_i)$ in $r_i$'s preference list, i.e. $rank_{r_i}(h_j) > rank_{r_i}(M(r_i))$. For each hospital $h_k$, let $r_p$ be the worst resident assigned to it in $M$. We then remove each resident $r_q$ from the preference list of $h_k$ and remove $h_k$ from the preference list of $r_q$ whenever $rank_{h_j}(r_q) > rank_{h_j}(r_p)$. The effect of these truncations is that each agent is assigned in $M$ to a "worst choice" partner. Let us call the instance with truncated preference lists $I'$, and its underlying graph $G'$, i.e. $G'$ contain only those edges representing the truncated preference lists of the agents. It is straightforward to see that it takes $O(m)$ time to construct $G'$.

By Lemma 3.3.1, $M$ is Pareto optimal if it admits no augmenting coalition or cyclic coalition. We can check for the former structure by testing for an augmenting path in $G'$. We use a similar form of restricted breadth-first search as described in Section 1.2, in

| Resident Pref | Hospital Pref | Hospital Capacity |
|---|---|---|
| $r_1$: $h_2$ $h_1$ | $h_1$: $r_1$ | $h_1 : 1$ |
| $r_2$: $h_2$ | $h_2$: $r_1$ $r_2$ | $h_2 : 1$ |

Figure 3.1: An instance $I_1$ of HR

which only edges not in $M$ are followed from vertices in $R$ and edges of $M$ are followed from vertices in $H$. If an augmenting path $P$ is found in $G'$, then by augmenting along $P$, we obtain a new matching $M'$ such that each resident $r_i$ prefers $M'(r_i)$ to $M(r_i)$, and each hospital $h_j$ either prefers $M'$ to $M$ or is indifferent between the two matchings as a result of the preference list truncations. Hence, $M$ is not Pareto optimal if $G'$ admits an augmenting path. It is straightforward that this takes $O(m)$ time.

To test for a cyclic coalition, we form a directed graph $G_C$ with respect to $M$ by letting $G_C$ consist of one vertex for each assigned resident in $I'$. We then construct an edge from a resident $r_i$ to another resident $r_j$ in $G_C$ if $r_i$ prefers $M(r_j)$ to $M(r_i)$ in $I'$. By a similar argument to the above, it follows that $M$ is cyclic-coalition-free if and only if $G_C$ is acyclic. Note that even though $M$ is a matching of a HR instance, all vertices in $G_C$ have only unitary capacity (being resident vertices). It follows that a depth-first search suffices to detect any cycles in $O(m)$ time so that these observations lead us to the following lemma.

**Lemma 3.3.2.** *Let $M$ be a matching in a given instance of* HR*. Then we may check whether $M$ is Pareto optimal in $O(m)$ time.*

Figure 3.1 shows us an HR instance $I_1$. Note that the unique stable matching $M = \{(r_1, h_2)\}$ has cardinality 1 here, but the maximum Pareto optimal matching $M' = \{(r_1, h_1), (r_2, h_2)\}$ has cardinality 2. Hence, Pareto optimality is a criterion that can give rise to larger matchings than stability. Furthermore, it is straightforward to see that by creating $p$ copies of $J$, we may construct an HR instance $J^p$ with $4p$ agents which admits a stable matching $M_p = \{(r_{2i+1}, h_{2i+2}) : 0 \leq i \leq p-1\}$ of size $p$ and a Pareto optimal matching $M'_p = (\{r_{2i+1}, h_{2i+1}), (r_{2i+2}, h_{2i+2}) : 0 \leq i \leq p-1\}$ of size $2p$. It follows that we can have an infinite family of HR instances for which the cardinality of a stable matching is half the size of a maximum Pareto optimal matching.

Hence, given any HR instance $I$, we are also interested in considering the problem of finding a maximum Pareto optimal matching in $I$ as an alternative optimality criterion. The next section presents an efficient algorithm for solving this.

## 3.4 Maximum Pareto optimal matchings in HR

In this section, we describe a two-phase algorithm for finding a maximum Pareto optimal matching in an instance $I$ of HR by satisfying the necessary and sufficient conditions of Lemma 3.3.1. Phase 1 of the algorithm uses Gabow's algorithm [15] to compute a maximum matching $M$ in the underlying graph $G$. This phase guarantees that $M$ admits no augmenting coalition and takes $O(\sqrt{C}m)$ time. Phase 2 transforms $M$ into a matching that admits no cyclic coalition in the following way.

Let us construct a graph $G'$ from $G$ by repeating the truncation of preference lists as described in the aforementioned methods for testing a given matching in $I$ for Pareto optimality. Hence, all agents are assigned to a worst-choice partner in $G'$. It thus follows as a result that the residents always improve and the hospitals either improve or remain indifferent even if we satisfy any cyclic coalition with respect to $M$ by considering preference lists on only one side. This allows us to obtain a cyclic-coalition-free matching in $G'$ from $M$ by considering the problem from only the point of view of the residents' truncated preference lists, which effectively transforms the problem in $I$ to an instance of the analogous problem for the Capacitated House Allocation problem.

Hence, this allows the Phase 3 algorithm, described in Chapter 2 for finding a maximum Pareto optimal matching given a CHA instance, to be reused from the residents' point of view in order to obtain a matching $M'$ from $M$ that is cyclic-coalition-free in $G$. We note that $M'$ must be cyclic-coalition-free by the correctness proof presented in Section 2.4.2.3. Furthermore, the correctness proof also shows that $M'$ remains a maximum matching since each resident who was assigned at the end of Phase 1 remains assigned after the execution of the Phase 3 algorithm. With the use of suitable data structures such as those described in Section 2.4.3, the Phase 3 algorithm is guaranteed to run in $O(m)$ time. Hence, it takes $O(m)$ time for Phase 2 of our algorithm to find a maximum Pareto optimal matching given a HR instance. It follows that Phase 1 dominates the overall complexity of our two-phase algorithm for HR, giving us the following result.

**Theorem 3.4.1.** *A maximum Pareto optimal matching in an instance $I$ of* HR *can be found in $O(\sqrt{C}m)$ time.*

## 3.5 Maximum Pareto optimal matchings in SMI

Since SMI is a special case of HR, it follows that Lemma 3.3.1 also gives a necessary and sufficient condition for a matching in an SMI instance $J$ to be Pareto optimal.

We first show how to test a given matching $M$ in $J$ for Pareto optimality. Let $G'$ be the underlying graph of $J$ and let $m$ be the number of edges in $G'$. We form a subgraph $G_M$ of $G'$ by letting $G_M$ contain only those edges in $M \cup bp(M)$ where $bp(M)$ is the set of blocking pairs with respect to $M$. It is clear that $G_M$ can be constructed in $O(m)$ time by considering the edges of $G$. By Lemma 3.3.1, $M$ is Pareto optimal in $J$ if and only if $M$ admits no augmenting path or alternating cycle in $G_M$. Clearly, we can test for an augmenting path in $G_M$ in $O(m)$ time using restricted breadth-first search as described in Section 1.2. In order to test for an alternating cycle, we remove any unmatched vertices and their incident edges from $G_M$. Any cycle in $G_M$ that remains is an even-length alternating cycle, so that a depth-first search suffices to detect this. Hence, we can check if a matching in $J$ is Pareto optimal in $O(m)$ time.

We next show how the algorithm for finding a maximum Pareto optimal matching in HR can be easily modified for the analogous problem in SMI. First of all, we use the Hopcroft-Karp algorithm for finding a maximum matching $M$ in Phase 1 instead of Gabow's algorithm. Then, this step takes $O(\sqrt{n}m)$ time where $n$ is the total number of men and women. Using a similar form of preference list truncation to that for HR as described above, we can then transform the problem in $J$ to the House Allocation problem (HA).

Hence, this allows Phase 3 of the implementation of the Top Trading Cycles Method in [3] to be reused from the men's point of view in order to obtain a matching $M'$ from $M$ that is cyclic-coalition-free in $G$. We note that $M'$ must be cyclic-coalition-free by the correctness of the Top Trading Cycles Method [57]. Furthermore $M'$ remains a maximum matching since the algorithm in [3] ensures that each man and woman who is assigned at the end of Phase 1 of our algorithm is also assigned when it completes execution. With the use of suitable data structures such as doubly linked lists or arrays, we can ensure that the initialisation and subsequent deletion of entries from the preference lists takes $O(m)$ time. Since the nested loops in Phase 3 of the implementation in [3] are guaranteed to run in $O(m)$ time, Phase 2 of our algorithm also runs in $O(m)$ time. This gives us the following result.

**Theorem 3.5.1.** *A maximum Pareto optimal matching in an instance $J$ of* SMI *can be found in $O(\sqrt{n}m)$ time.*

It follows that it is generally faster to find a maximum Pareto optimal matching in an SMI instance using the techniques described in this section as opposed to the algorithm for HR described in Section 3.4.

## 3.6 Open problems

In this chapter, we presented efficient algorithms for finding a maximum Pareto optimal matching given an instance of HR or SMI. A number of open problems remain. For example,

- The problem of finding a maximum Pareto optimal matching given an instance of HR or SMI can be generalised by permitting agents to contain ties in their preference lists, i.e. HRT or SMTI respectively. As with CHA, the definition of the relation $\prec$ remains unchanged in the context of ties. For SMTI, we remark that a maximum Pareto optimal matching can be found in $O(nm + n^2 \log n)$ time [14] by transformation to the Assignment problem in an analogous way to that described in Section 2.2 for the case of CHA. However, it is open as to whether a similar form of transformation exists to solve the problem in HRT. It is also open as to whether the problem of finding a maximum Pareto optimal matching, given an instance of HRT or SMTI, is also solvable in the same time complexity as its counterpart in the case of strict preferences?

- [7] shows that the problem of finding a maximum Pareto optimal matching in an instance of SRI is solvable in $O(\sqrt{n\alpha(m,n)}m \log^{3/2} n)$ time. However, it remains to consider whether a maximum Pareto optimal matching in SRI can be constructed in $O(\sqrt{n}m)$ time, which is the complexity of the fastest current algorithm for finding a maximum matching in a general graph [44].

# Chapter 4

# Popular matchings in CHAT

## 4.1 Introduction

In this chapter, we consider the problem of finding a maximum popular matching given an instance of CHA or CHAT. As noted in Section 1.3.2.2, many recent papers have focused on popular matchings, given the importance of voting to many economic decisions, and also given the viability of popular matchings as an optimality criterion for matching problems with one-sided preferences. Here, we present the first algorithmic results for computing popular matchings in a capacitated bipartite graph. The main results of this chapter, and their organisation are as follows.

In Section 4.2, we first develop a characterisation of popular matchings in a CHA instance $I$. We then use this characterisation to construct an $O(\sqrt{C}n_1 + m)$ algorithm for finding a maximum popular matching in $I$ or reporting that none exists, where $C$ is the total capacity of the houses, $n_1$ is the total number of agents and $m$ is the total length of the preference lists in $I$ respectively. In Section 4.3, we provide the first extension of the Edmonds-Gallai Decomposition to the case of a capacitated bipartite graph. Using this, we build a new characterisation of popular matchings in a CHAT instance $J$, and then use it to construct an $O(\sqrt{C}m)$ algorithm for finding a maximum popular matching in $J$ or reporting that none exists. We finally remark that a straightforward solution to each of the problems of finding a maximum popular matching, given an instance $I$ of CHA or CHAT, may be to use "cloning". Informally, this entails creating $c_j$ clones for each house $h_j$, to obtain an instance $C(I)$ of HAT (i.e. each house has capacity 1), and then applying the algorithms of [4] to $C(I)$. However, we will show in Sections 4.2 and 4.3 that this method in general leads to slower algorithms than the direct approach that we will

be using in each case.

## 4.2 Popular matchings in CHA

### 4.2.1 Characterising popular matchings

Let $I$ be an instance of CHA, and let $G = (A, H, E)$ be the underlying graph of $I$, where $A = \{a_1, a_2, ..., a_{n_1}\}$ is the set of *agents*, $H = \{h_1, h_2, ..., h_{n_2}\}$ is the set of *houses* and $E$ is the set of edges in $G$ representing the acceptable houses of the agents. We assume all the notations and terminology that have been defined for CHA in Chapters 1 and 2. Recall that, given two matchings $M$ and $M'$ in $I$, we say that an agent $a_i$ *prefers* $M'$ to $M$ if either (i) $a_i$ is assigned in $M'$ and unassigned in $M$, or (ii) $a_i$ is assigned in both $M'$ and $M$ and prefers $M'(a_i)$ to $M(a_i)$. Let $P(M', M)$ denote the set of agents who prefer $M'$ to $M$. Then, $M'$ is *more popular than* $M$ if $|P(M', M)| > |P(M, M')|$, i.e. the number of agents who prefer $M'$ to $M$ is greater than the number of agents who prefer $M$ to $M'$. Furthermore, a matching $M$ in $I$ is *popular* if there is no other matching $M'$ in $I$ that is more popular than $M$.

For each agent $a_i \in A$, let $f(a_i)$ denote the first-ranked house on $a_i$'s preference list. Any such house $h_j$ is called an *$f$-house*. For each $h_j \in H$, let $f(h_j) = \{a_i \in A : f(a_i) = h_j\}$ and $f_j = |f(h_j)|$ (possibly $f_j = 0$). We create a unique *last resort* house $l(a_i)$ with capacity 1 for each agent $a_i \in A$, and append $l(a_i)$ to $a_i$'s preference list. We also henceforth assume that $G$ contains the vertex $l(a_i)$ and the edge $(a_i, l(a_i))$ for each $a_i \in A$, and that $H$ contains the respective last resort houses. We let $n = n_1 + n_2$ and $m = |E|$.

The following lemma is a vital first step in characterising popular matchings in $I$.

**Lemma 4.2.1.** *Let $M$ be a popular matching in $I$. Then for every $f$-house $h_j$, $|M(h_j) \cap f(h_j)| = \min\{c_j, f_j\}$.*

*Proof.* We consider the following two cases.

– *Case (i):* Suppose $f_j \leq c_j$. We will show that $f(h_j) \subseteq M(h_j)$. For, suppose not. Then choose any $a_r \in f(h_j) \backslash M(h_j)$. We consider the subcases that (a) $h_j$ is undersubscribed and (b) $h_j$ is full. In subcase (a), promote $a_r$ to $h_j$ to obtain a more popular matching than $M$. In subcase (b), choose any $a_s \in M(h_j) \backslash f(h_j)$. Let $h_k = f(a_s)$. Then $h_k \neq h_j$. If $h_k$ is undersubscribed, promote $a_r$ to $h_j$ and promote $a_s$ to $h_k$ to obtain a more popular matching than $M$. Otherwise, choose any $a_t \in M(h_k)$. If $a_r = a_t$, we

then promote $a_r$ to $h_j$ and promote $a_s$ to $h_k$ to obtain a more popular matching than $M$. Otherwise, we then promote $a_r$ to $h_j$, promote $a_s$ to $h_k$ and demote $a_t$ to $l(a_t)$ to obtain a more popular matching than $M$.

– *Case (ii):* Suppose $f_j > c_j$. If $h_j$ is undersubscribed, then $f(h_j) \not\subseteq M(h_j)$ so there exists some $a_r \in f(h_j) \backslash M(h_j)$ that we can promote to $h_j$ to obtain a more popular matching as in Case (i)(a). Hence, $h_j$ is full. Now, suppose for a contradiction that $M(h_j) \not\subseteq f(h_j)$. Then there exists some $a_s \in M(h_j) \backslash f(h_j)$. As $f_j > c_j$, it follows that $f(h_j) \not\subseteq M(h_j)$ so there exists some $a_r \in f(h_j) \backslash M(h_j)$. The remainder of the argument follows Case (i)(b).

Hence the following properties hold for the new matching. If $f_j \leq c_j$, then $f(h_j) \subseteq M(h_j)$. Otherwise, $M(h_j) \subseteq f(h_j)$ and $|M(h_j)| = c_j$. Thus, the condition in the statement of the lemma is now satisfied. □

For each agent $a_i$, we next define $s(a_i)$ to be the most-preferred house $h_j$ on $a_i$'s preference list such that either (i) $h_j$ is a not an $f$-house, or (ii) $h_j$ is an $f$-house such that $h_j \neq f(a_i)$ and $f_j < c_j$. Note that $s(a_i)$ must exist in view of $l(a_i)$. We refer to such a house $h_j$ as an *s-house*. We remark that the set of $f$-houses need not be disjoint from the set of $s$-houses. It may be shown that a popular matching $M$ will only assign an agent $a_i$ to either $f(a_i)$ or $s(a_i)$, as indicated by the next two lemmas.

**Lemma 4.2.2.** *Let $M$ be a popular matching in $I$. Then no agent $a_i \in A$ can be assigned in $M$ to a house between $f(a_i)$ and $s(a_i)$ on $a_i$'s preference list.*

*Proof.* Suppose that $a_i$ is assigned to a house $h_k$ between $f(a_i)$ and $s(a_i)$. Then $h_k$ is an $f$-house and $f_k \geq c_k$, for otherwise $s(a_i) = h_k$. As $f_k \geq c_k$, by Lemma 4.2.1, $M(h_k) \subseteq f(h_k)$. However, $f(a_i) \neq h_k$, thus $a_i \notin f(h_k)$. Hence, $a_i$ cannot be assigned to $h_k$. □

**Lemma 4.2.3.** *Let $M$ be a popular matching in $I$. Then no agent $a_i \in A$ can be assigned in $M$ to a house worse than $s(a_i)$ on $a_i$'s preference list.*

*Proof.* Let $h_j = s(a_i)$. If $h_j$ is undersubscribed, then we can promote $a_i$ to $h_j$, a contradiction. Hence, $h_j$ is full. We consider two cases.

– *Case (i):* $h_j$ is an $f$-house. By definition of an $s$-house, $f_j < c_j$, so there exists some $a_r \in M(h_j) \backslash f(h_j)$. Let $h_k = f(a_r)$. Then $h_k \neq h_j$. As $c_k \geq 1$ and $f_k \geq 1$, it follows by Lemma 4.2.1 that $M(h_k) \neq \emptyset$. Let $a_s \in M(h_k)$. Now, if $a_i = a_s$, we can then promote $a_i$ to $h_j$ and promote $a_r$ to $h_k$ to obtain a more popular matching than $M$, a contradiction.

Otherwise, we can then promote $a_i$ to $h_j$, promote $a_r$ to $h_k$, and demote $a_s$ to $l(a_s)$ to obtain a more popular matching than $M$, a contradiction.

– *Case (ii):* $h_j$ is not an $f$-house. Let $a_r \in M(h_j)$. Then $a_r \notin f(h_j)$. The remainder of the proof of this case proceeds as in Case (i). □

Recall that $G$ is the underlying graph of $I$. We form a subgraph $G'$ of $G$ by letting $G'$ contain only two edges for each agent $a_i$, that is, one to $f(a_i)$ and the other to $s(a_i)$. We say that a matching $M$ is *agent-complete* in a given graph if it assigns all agents in the graph. It follows that, in view of last resort houses, all popular matchings must be agent-complete in $G'$. However, $G'$ need not admit an agent-complete matching if $s(a_i) \neq l(a_i)$ for some agent $a_i$. In conjunction with Lemmas 4.2.1-4.2.3, the graph $G'$ gives rise to the following characterisation of popular matchings in $I$.

**Theorem 4.2.1.** *A matching $M$ is popular in $I$ if and only if*

1. *for every $f$-house $h_j$,*
   (a) *if $f_j \leq c_j$, then $f(h_j) \subseteq M(h_j)$;*
   (b) *if $f_j > c_j$, then $|M(h_j)| = c_j$ and $M(h_j) \subseteq f(h_j)$.*
2. *$M$ is an agent-complete matching in the reduced graph $G'$.*

*Proof.* By Lemmas 4.2.1-4.2.3, any popular matching necessarily satisfies Conditions 1 and 2. We now show that these conditions are sufficient.

Let $M$ by any matching satisfying Conditions 1 and 2 and suppose for a contradiction that $M'$ is a matching that is more popular than $M$. Let $a_i$ be any agent that prefers $M'$ to $M$ and let $h_k = M'(a_i)$. Since $M$ is an agent-complete matching in $G'$, and since $G'$ contains only edges from $a_i$ to $f(a_i)$ and $s(a_i)$, then $M(a_i) = s(a_i)$. Hence either (i) $h_k = f(a_i)$ or (ii) $h_k$ is an $f$-house such that $h_k \neq f(a_i)$ and $f_k \geq c_k$, by definition of $s(a_i)$.

In Case (i), if $f_k < c_k$ then by Condition 1(a), $a_i \in M(h_k)$, a contradiction. Hence in both Cases (i) and (ii), $f_k \geq c_k$. In each of the cases that $f_k = c_k$ and $f_k > c_k$, it follows by Conditions 1(a) and 1(b) that $|M(h_k)| = c_k$ and $M(h_k) \subseteq f(h_k)$. Since $h_k$ is full in $M$, it follows that $|M(h_k) \backslash M'(h_k)| \geq |M'(h_k) \backslash M(h_k)|$. Hence for every $a_i$ who prefers $M'(a_i) = h_k$ to $M(a_i)$, there is a unique $a_j \in M(h_k) \backslash M'(h_k)$. But as $a_j \in M(h_k)$, it follows that $h_k = f(a_j)$. Hence $a_j$ prefers $M(a_j)$ to $M'(a_j)$. Therefore, $M$ is popular in $I$. □

**Algorithm 6** Algorithm Popular-CHA

1: $M := \emptyset$;
2: **for each** $f$-house $h_j$ **do**
3: $c'_j := c_j$;
4: **if** $f_j \leq c_j$ **then**
5: **for each** $a_i \in f(h_j)$ **do**
6: $M := M \cup \{(a_i, h_j)\}$;
7: delete $a_i$ and its incident edges from $G'$;
8: $c'_j := c_j - f_j$;
9: remove all isolated and full houses, and their incident edges, from $G'$;
10: compute a maximum matching $M'$ in $G'$ using capacities $c'_j$;
11: **if** $M'$ is not agent-complete in $G'$ **then**
12: **output** "no popular matching exists"
13: **else**
14: $M := M \cup M'$;
15: **for each** $a_i \in A$ **do**
16: $h_j := f(a_i)$;
17: **if** $f_j > c_j$ **and** $|M(h_j)| < c_j$ **and** $h_j \neq M(a_i)$ **then**
18: promote $a_i$ from $M(a_i)$ to $h_j$ in $M$;

### 4.2.2 Finding a popular matching

Theorem 4.2.1 leads to Algorithm Popular-CHA for finding a popular matching in a CHA instance $I$, or reporting that none exists, as shown in Algorithm 6. The algorithm begins by using a pre-processing step (lines 2-9) on $G'$ that assigns agents to their first-choice house $h_j$ whenever $f_j \leq c_j$, so as to satisfy Condition 1(a) of Theorem 4.2.1.

Our next step computes a maximum matching $M'$ in $G'$, according to the adjusted house capacities $c'_j$ that are defined following pre-processing. We use Gabow's algorithm [15] to compute $M'$ in $G'$ and then test whether $M'$ is agent-complete. The pre-allocations are then added to $M'$ to give $M$. As a last step, we ensure that $M$ also meets Condition 1(b) of Theorem 4.2.1. For, suppose that $h_j \in H$ is an $f$-house such that $f_j > c_j$. Then by definition, $h_j$ cannot be an $s$-house. Thus if $a_k \in M(h_j)$ prior to the third for loop, it follows that $a_k \in f(h_j)$. At this stage, if $h_j$ is undersubscribed in $M$, we repeatedly promote any agent $a_i \in f(h_j) \backslash M(h_j)$ from $M(a_i)$ (note that $M(a_i)$ must be $s(a_i)$ and hence cannot be an $f$-house $h_l$ such that $f_l > c_l$) to $h_j$ until $h_j$ is full, ensuring that $M(h_j) \subseteq f(h_j)$.

It is clear that the reduced graph $G'$ of $G$ can be constructed in $O(m)$ time. The graph $G'$ has $O(n_1)$ edges since each agent has degree 2 in $G'$. It is straightforward to see that each of the pre- and post-processing steps involving the three for loop phases takes $O(n_1 + n_2)$ time. The complexity of Gabow's algorithm [15] for computing $M'$ in $G'$ is $O(\sqrt{C}n_1)$. Hence we obtain the following result concerning the complexity of Algorithm Popular-CHA.

**Lemma 4.2.4.** *Given an instance of* CHA*, we can find a popular matching, or determine that none exists, in* $O(\sqrt{C}n_1 + m)$ *time.*

### 4.2.3 Finding a maximum popular matching

It remains to consider the problem of finding a maximum popular matching in $I$. We begin by dividing the set of all agents into disjoint sets. Let $A_1$ be the set of all agents such that if $a_i$ is an agent in $A_1$, then $s(a_i) = l(a_i)$. Also, let $A_2 = A - A_1$. We aim to find a matching $M$ that satisfies the conditions of Theorem 4.2.1, and that minimises the number of $A_1$-agents who are assigned to their last resort house.

We begin by constructing $G'$, and carrying out the pre-processing step in lines 2-9 of Algorithm Popular-CHA on all agents in $A_1 \cup A_2$. We then try to find a maximum matching $M'$ in $G'$ that only involves the $A_2$-agents that remain after pre-processing and their incident edges. If $M'$ is not an agent-complete matching of the agents in $A_2$ that remain after pre-processing, then $G$ admits no popular matching by Theorem 4.2.1. Otherwise, we remove all edges in $G'$ that are incident to a last resort house, and try to assign $A_1$-agents to their first-choice houses. At each step, we try to assign an additional $A_1$-agent to his first-choice house by finding an augmenting path with respect to $M'$ using Gabow's algorithm [15], so that we have a maximum matching of agents in $A_1 \cup A_2$ in $G'$ at the end of this process. If any $A_1$-agent remains unassigned, we simply assign him to his last resort house, to obtain an agent-complete matching in $G'$. We also ensure that Condition 1(b) of Theorem 4.2.1 is met by executing the third for loop in Algorithm Popular-CHA. It follows that the matching so obtained, together with the pre-assignments from earlier, is a maximum popular matching, giving the following theorem.

**Theorem 4.2.2.** *Given an instance of* CHA*, we can find a maximum popular matching, or determine that none exists, in* $O(\sqrt{C}n_1 + m)$ *time.*

### 4.2.4 "Cloning" verus our direct approach

An alternative approach to our algorithm may be to use cloning. Given an instance $I$ of CHA, we may obtain an instance $J$ of HAT by creating $c_j$ clones $h_j^1, h_j^2, ..., h_j^{c_j}$ of each house $h_j$ in $I$, where each clone has a capacity of 1. In addition, we replace each occurrence of $h_j$ in a given agent's preference list with the sequence $h_j^1, h_j^2, ..., h_j^{c_j}$, the elements of which are listed in a single tie at the point where $h_j$ appears. We may then apply the $O(\sqrt{n}m)$ algorithm for HAT given by [4] to $J$ in order to find a maximum popular matching in $I$.

We now compare the worst-case complexity of the above cloning approach with that of our direct algorithm. The underlying graph $G_J$ of $J$ contains $n' = n_1 + C$ vertices. Let $c_{min} = \min\{c_j : h_j \in H\}$, and for $a_i \in A$, let $A_i$ denote the set of acceptable houses for $a_i$. Then the number of edges in $G_J$ is $m' = \sum_{a_i \in A} \sum_{h_j \in A_i} c_j \geq mc_{min}$. Hence the complexity of applying the algorithm given by [4] to $J$ is $\Omega(\sqrt{n_1 + C}mc_{min})$. Recall that the complexity of Algorithm Popular-CHA is $O(\sqrt{C}n_1 + m)$. It follows that the cloning method is slower by a factor of $\Omega(\sqrt{n_1 + C}c_{min})$ or $\Omega(mc_{min}/n_1)$ (note that $m \geq n_1$ and $c_{min} \geq 1$) according as $\sqrt{C}n_1 \leq m$ or $\sqrt{C}n_1 > m$ respectively.

## 4.3 Popular matchings in CHAT

In this section, we generalise the characterisation of popular matchings together with Algorithm Popular-CHA as given in the previous section to the case where we are given an instance of CHAT.

### 4.3.1 Characterising popular matchings

Let $M$ be a popular matching in an instance $I$ of CHAT. For each agent $a_i \in A$, let $f(a_i)$ denote the set of first-ranked houses on $a_i$'s preference list (clearly it is possible that $|f(a_i)| > 1$ in view of ties in the preference lists). We refer to all such houses $h_j$ as *f-houses* and we let $f(h_j) = \{a_i \in A : h_j \in f(a_i)\}$. Let $G = (A, H, E)$ be the underlying graph of $I$. Define $E_1 = \{(a_i, h_j) : a_i \in A \wedge h_j \in f(a_i)\}$ to be the set of *first-choice edges*. We define the *first-choice graph* of $G$ as $G_1 = (A, H, E_1)$.

Given a CHAT instance $I$, since it is possible for an agent to have greater than one $f$-house, Lemma 4.2.1 no longer holds in general. For example, it is possible for an $f$-house $h_j$ such that $f_j = c_j$ to not be assigned to all the agents in $f(h_j)$ in a popular matching

if there are more $f$-houses than agents in $I$. This makes $h_j$ eligible to be the $s$-house of some agent not in $f(h_j)$ in $I$ whereas this would not have been possible in any given CHA instance. Hence, we will work towards a new definition of $s$-houses in the context of CHAT in this subsection. For instances with strict preference lists, Lemma 4.2.1 implies that $M \cap E_1$ is a maximum matching in $G_1$. As the next lemma indicates, this latter condition also extends to the CHAT case.

**Lemma 4.3.1.** *Let $M$ be a popular matching in $I$. Then $M \cap E_1$ is a maximum matching in $G_1$.*

*Proof.* Let $M_1 = M \cap E_1$. Suppose for a contradiction that $M_1$ is not a maximum matching in $G_1$. Then $M_1$ admits an augmenting path $P = \langle a_1, h_1, ..., a_k, h_k \rangle$ with respect to $G_1$. Now, in view of last resort houses, $a_1$ must be assigned in $M$. It follows that $M(a_1) \notin f(a_1)$, for otherwise $M(a_1) \in P$. We let $M' = M \setminus \{(a_1, M(a_1))\}$. We consider the following cases for $h_k$.

– *Case (i):* $h_k$ is undersubscribed in $M'$. As $a_1$ is unassigned in $M'$, $h_1 \in f(a_1)$, and $|M'(h_k)| < c_k$, we can augment $M'$ with $P$ to obtain a new matching $M''$. Then, $a_1$ is assigned with $h_1$ in $M''$. Furthermore, as all edges in $G'$ are first-choice edges, all other agents in $P$ become assigned in $M''$ to one of their other first-choice houses. However, $M''$ is more popular than $M$, a contradiction.

– *Case (ii):* $|M'(h_k)| = c_k$. Choose any $a_s \in M'(h_k) \setminus f(h_k)$. Note that such an $a_s$ must exist, for $h_k$ is full in $M'$ but undersubscribed in $M_1$. Clearly, $a_s \neq a_i$ for $1 \leq i \leq k$. Choose any $h_t \in f(a_s)$. Now, $h_t$ cannot be a house in $P$ for suppose not. Without loss of generality, let $h_t = h_j$ where $1 \leq j < k$. Let $C = \langle h_j, a_{j+1}, h_{j+1}, ..., a_k, h_k, a_s \rangle$. Let also $M'' = M' \oplus C$. It follows that each agent $a_x \neq a_s$ in $C$ becomes assigned in $M''$ to one of their other first-choice houses while $a_s$ improves by becoming assigned in $M''$ to one of his first-choice house. However then, $M''$ is more popular than $M$, a contradiction. Hence, $h_t$ does not belong to $P$. Now, if $h_t$ is undersubscribed in $M'$, we can then augment $M'$ with $P$ so that $a_1$ improves and the other agents in $P$ are indifferent, and promote $a_s$ to $h_t$ to obtain a more popular matching than $M$, a contradiction. Otherwise, choose any $a_u \in M'(h_t)$. Since $h_t \notin P$ and $a_1$ is unassigned in $M'$, $a_u$ is a distinct agent from any agent in $P$. However, we can then augment $M'$ with $P$ so that $a_1$ improves and the other agents in $P$ are indifferent, promote $a_s$ to $h_t$ and demote $a_u$ to $l(a_u)$ to obtain a more popular matching than $M$, a contradiction. □

We now work towards a definition of $s$-houses by using the Edmonds-Gallai Decomposition. Let $M$ be a maximum matching in some bipartite graph $G$ where all vertices have capacity 1. According to Lemma 1.2.1, the vertices of $G$ can be partitioned into three disjoint sets: $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$. Vertices in $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ are called *even, odd,* and *unreachable* respectively. A vertex $v$ is even (odd) if there exists an alternating path of even (odd) length from an unassigned vertex in $G$ to $v$. If no such alternating path exists, $v$ is unreachable. As noted in Section 1.2, this vertex labelling is also known as the EOU labelling. Our aim is to obtain an EOU labelling of $G_1$ relative to a maximum matching $M_1$ of $G_1$ (as obtained by Gabow's algorithm [15], for example). However Lemma 1.2.1 applies directly only to the case where each vertex in the given bipartite graph has capacity 1. We will show that the Edmonds-Gallai Decomposition also holds in the case of a capacitated bipartite graph as follows.

Let $G = (U, W, E)$ be a capacitated bipartite graph. Also, let $M$ be a maximum matching of $G$. Let $C(G)$ be a *cloned graph* of $G$ by replacing every vertex $w_j \in W$ with the clones $w_j^1, w_j^2, \ldots, w_j^{c_j}$ where $c_j$ is the capacity of $w_j$. We then divide the capacity of each vertex $w_j \in W$ among its clones by allowing each clone to have capacity 1. In addition, if $(u_i, w_j)$ belongs to $G$, then we add $(u_i, w_j^k)$ to $C(G)$ for all $k$ $(1 \leq k \leq c_j)$. We then adapt the maximum matching $M$ in $G$ to obtain a matching $C(M)$ in $C(G)$, as follows. If a vertex $w_j \in W$ in $G$ is assigned to $x_j$ vertices $u_1, ...u_{x_j}$ in $M$, then we add $(u_k, w_j^k)$ to $C(M)$ for $1 \leq k \leq x_j$, so that $|C(M)| = |M|$ and $C(M)$ is a maximum matching in $C(G)$. It follows that $C(G)$ is a bipartite graph in which all of its vertices on the right hand side have capacity 1.

Let us now clone $C(G)$ to obtain a bipartite graph $C(G)'$ in which all of its vertices have capacity 1 by repeating the above steps for the vertices in $U$. That is, we replace every vertex $u_i \in U$ with the clones $u_i^1, u_i^2, \ldots, u_i^{c_i}$ where $c_i$ is the capacity of $u_i$. We then divide the capacity of each vertex $u_i \in U$ among its clones by allowing each clone to have capacity 1. In addition, if $(u_i, w_j^k)$ belongs to $C(G)$, then we add $(u_i^l, w_j^k)$ to $C(G)$ for all $l$ $(1 \leq l \leq c_i)$ where $w_j^k$ is a clone of $w_j \in W$ in $C(G)$. We then adapt the maximum matching $C(M)$ in $C(G)$ to obtain a matching $C(M)'$ in $C(G)'$ as follows. If a vertex $u_i \in U$ in $G$ is assigned to $y_i$ vertices $w'_1, ...w'_{y_i}$ in $C(M)$, then we add $(u_i^k, w'_k)$ to $C(M)'$ for $1 \leq k \leq y_i$ where without loss of generality, $w'_k$ is a clone of the vertex $w_k \in W$ in $C(G)$. It follows that $|C(M)'| = |M|$ and $C(M)'$ is a maximum matching in $C(G)'$. It also follows that $C(G)'$ is a bipartite graph in which all of its vertices have capacity 1.

Suppose that we are given an EOU labelling of the vertices in $C(G)'$ with respect to $C(M)'$ based on Lemma 1.2.1. The next lemma shows that the clones corresponding to each vertex $v_j \in U \cup W$ in $G$ have the same EOU label in $C(G)'$.

**Lemma 4.3.2.** *Let $G$ be a capacitated bipartite graph and let $M$ be a maximum matching in $G$. Define the cloned graph $C(G)'$ and its corresponding maximum matching $C(M)'$ as above. Then, given any vertex $v_j \in U \cup W$, any two clones of $v_j$ in $C(G)'$ have the same EOU label.*

*Proof.* Without loss of generality, let $v_j$ be a vertex $w_j$ belonging to the vertex set $W$; analogous results can be proven if $v_j \in U$. Let $w_j^x$ and $w_j^y$ be two clones corresponding to $w_j$. We consider the cases where (1) $w_j^x$ is even, (2) $w_j^x$ is odd, and (3) $w_j^x$ is unreachable.

Case (1): If $w_j^x$ is even, then we consider the subcases where (a) $w_j^x$ is assigned in $C(M)'$, and (b) $w_j^x$ is unassigned in $C(M)'$. In subcase (a), if $w_j^y$ is unassigned in $C(M)'$, then it follows immediately that $w_j^y$ is also even. Hence, suppose that $w_j^y$ is also assigned in $C(M)'$. As $w_j^x$ is even, there exists an even length alternating path $P$ to $w_j^x$ in $C(G)'$ from an unassigned vertex clone belonging to $W$. Let $u_i^p$ be the vertex that precedes $w_j^x$ on $P$ where $u_i^p$ is a clone vertex of $u_i$ in $C(G)'$. It follows that $(u_i^p, w_j^x) \in C(M)'$ from our definition of the path $P$. As $w_j^y$ is also assigned in $C(M)'$, let $(u_k^q, w_j^y) \in C(M)'$ where $u_k^q$ is a clone vertex of $u_k$ in $C(G)'$ and $(u_k, w_j) \in M$. Then, it follows that $(u_k^q, w_j^x)$ must be an edge in $C(G)'$. As $w_j^x$ is even, $u_k^q$ is odd. As a result, $w_j^y$ is even. In subcase (b), if $w_j^y$ is also unassigned in $C(M)'$, then it is again immediate that $w_j^y$ is also even. Hence, suppose that $w_j^y$ is assigned in $C(M)'$, to $u_i^p$ say, where $u_i^p$ is a clone vertex of $u_i$ in $C(G)'$ and $(u_i, w_j) \in M$. Now $(u_i^p, w_j^x)$ is also an edge in $C(G)'$. As $w_j^x$ is even, $u_i^p$ is odd, and hence $w_j^y$ is even.

Case (2): If $w_j^x$ is odd, then there must exist an odd-length alternating path from an unassigned vertex $u_i^p$ to $w_j^x$ where $u_i^p$ is a clone of $u_i \in U$ in $C(G)'$. It follows that $w_j^x$ cannot be unassigned for otherwise $C(M)'$ admits an augmenting path, a contradiction. Hence, $w_j^x$ is assigned in $C(M)'$ to $u_k^q$, say, where $u_k^q$ is a clone of $u_k \in U$ in $C(G)'$ and $(u_k, w_j) \in M$. Then, $u_k^q$ is even. However, $(u_k^q, w_j^y)$ is an edge in $C(G)'$, so it follows that $w_j^y$ is odd.

Case (3): Now, $w_j^y$ must also be unreachable. For, suppose not. If $w_j^y$ is even, then $w_j^x$ is also even by Case (1), a contradiction. If $w_j^y$ is odd, then $w_j^x$ is also odd by Case (2), a contradiction. □

In view of Lemma 4.3.2, it follows that the clones corresponding to each vertex $v_j \in U \cup W$ have the same EOU label in $C(G)'$, thereby giving us a well-defined characterisation of EOU labels of all vertices in $G$. That is, if the clones of the vertex $v_j$ are even, odd or unreachable in $C(G)'$, we can correspondingly label $v_j$ as even, odd or unreachable in $G$. Suppose that we now have an EOU labelling of the vertices in $G$ as described above. The next result is a consequence of Lemma 4.3.2.

**Lemma 4.3.3.** *Let $G = (U, W, E)$ be a capacitated bipartite graph and let $M$ be a maximum matching in $G$. Then every odd or unreachable vertex $v_j \in U \cup W$ satisfies $|M(v_j)| = c_j$.*

*Proof.* Let $v_j \in U \cup W$ be any vertex that is odd (or unreachable) in $G$. By Lemma 4.3.2, all clones of $v_j$ will also be odd (or unreachable) in $C(G)'$. It follows that $v_j$ must be full in $M$, for otherwise, at least one of its clones $v_j^x$ will be unassigned in $C(M)'$. However, $v_j^x$ will then be even, a contradiction. □

Hence, Lemma 4.3.2 and Lemma 4.3.3 give rise to the first extension of the Edmonds-Gallai Decomposition to the capacitated bipartite graph as follows.

**Lemma 4.3.4.** *Let $G = (U, W, E)$ be a capacitated bipartite graph and let $M$ be a maximum matching in $G$. Define $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ to be the vertex sets corresponding to even, odd and unreachable vertices in an EOU labelling of $G$ with respect to $M$. Then:*

*(a) The sets $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ are pairwise disjoint. Every maximum matching in $G$ partitions the vertices into the same sets of even, odd and unreachable vertices.*

*(b) Every maximum matching $M$ in $G$ satisfies the following properties:*

*(i) every vertex in $\mathcal{O}$ and every vertex in $\mathcal{U}$ is full in $M$;*

*(ii) every vertex in $\mathcal{O}$ is assigned only to vertices in $\mathcal{E}$ in $M$;*

*(iii) every vertex in $\mathcal{U}$ is assigned only to vertices in $\mathcal{U}$ in $M$;*

*(iv) $|M| = \sum_{u_i \in \mathcal{O}_U} c_i + \sum_{w_j \in \mathcal{O}_W} c_j + \sum_{u_i \in \mathcal{U}_U} c_i$ where $\mathcal{O}_U$ is the set of odd vertices in $U$, $\mathcal{O}_W$ is the set of odd vertices in $W$, and $\mathcal{U}_U$ is the set of unreachable vertices in $U$.*

*(c) No maximum matching in $G$ contains an edge between two vertices in $\mathcal{O}$ or a vertex in $\mathcal{O}$ with a vertex in $\mathcal{U}$. There is no edge in $G$ connecting a vertex in $\mathcal{E}$ with a vertex in $\mathcal{U}$, or between two vertices of $\mathcal{E}$.*

It follows that Lemma 4.3.4 enables us to obtain an EOU labelling of $G_1$ relative to a maximum matching $M_1$ of $G_1$. The following corollary is a result of Lemma 4.3.4.

**Corollary 4.3.1.** *Let $M$ be a popular matching in $I$. Then every odd or unreachable house $h_j \in H$ satisfies $M(h_j) \subseteq f(h_j)$.*

*Proof.* Let $G_1 = (A, H, E_1)$ be the first-choice graph of $I$. Then, $M_1 = M \cap E_1$ is a maximum matching in $G_1$ by Lemma 4.3.1. Let $h_j \in H$ be any house that is odd (or unreachable) in $G_1$. By Lemma 4.3.4, $h_j$ is full in $M_1$. Since $C(G_1)$ contains only first-choice edges, it follows that $M_1(h_j) \subseteq f(h_j)$, and hence $M(h_j) \subseteq f(h_j)$. □

We are now in a position to define $s(a_i)$, the set of houses such that, in a popular matching $M$, if $a_i \in A$ is assigned in $M$ and $M(a_i) \notin f(a_i)$, then $M(a_i) \in s(a_i)$. We will ensure that any odd or unreachable house $h_j$ is not a member of $s(a_i)$, since $|M(h_j)| = c_j$ and $M(h_j) \subseteq f(h_j)$ by Lemma 4.3.4 and Corollary 4.3.1. Hence, we define $s(a_i)$ to be the set of highest-ranking houses in $a_i$'s preference list that are even in $G_1$. Any such house is called an *s-house*. Clearly, it is possible that $|s(a_i)| > 1$, however, $a_i$ is indifferent between all houses in $s(a_i)$. Furthermore, $s(a_i) \neq \emptyset$ due to the existence of last resort houses which are of degree 0 in $G_1$ (and thus even). However, $f(a_i)$ and $s(a_i)$ need not be disjoint, i.e. either $f(a_i) = s(a_i)$ or $a_i$ prefers all members of $f(a_i)$ to $s(a_i)$. It turns out that Lemmas 4.2.2 and 4.2.3 also extend to CHAT as established by the following lemmas.

**Lemma 4.3.5.** *Let $M$ be a popular matching in $I$. Then no agent $a_i \in A$ can be assigned in $M$ to a house between $f(a_i)$ and $s(a_i)$ on $a_i$'s preference list.*

*Proof.* Suppose that $a_i$ is assigned to a house $h_j$ strictly between $f(a_i)$ and $s(a_i)$. Then, $a_i$ must prefer $h_j$ to all houses in $s(a_i)$. Hence, $h_j$ must be an odd or unreachable house in $G_1$, as $s(a_i)$ contains the highest-ranking even houses in $G_1$ in $a_i$'s preference list. By Corollary 4.3.1, $M(h_j) \subseteq f(h_j)$. However, this is a contradiction as $h_j \notin f(a_i)$. □

**Lemma 4.3.6.** *Let $M$ be a popular matching in $I$. Then no agent $a_i \in A$ can be assigned in $M$ to a house worse than $s(a_i)$ on $a_i$'s preference list.*

*Proof.* Suppose that $a_i$ is assigned to a house worse than $s(a_i)$. Let $h_j$ be any house in $s(a_i)$. Now, if $|M(h_j)| < c_j$, we can promote $a_i$ to $h_j$ to obtain a more popular matching. Hence, suppose that $|M(h_j)| = c_j$. Let $a_k \in M(h_j)$. We consider two cases for $h_j$.
– *Case (i):* $h_j \notin f(a_k)$. We then choose any $h_l \in f(a_k)$. If $|M(h_l)| < c_l$, we promote

$a_i$ to $h_j$ and promote $a_k$ to $h_l$ to obtain a more popular matching than $M$. Otherwise, $|M(h_l)| = c_l$ so we let $a_m \in M(h_l)$. If $a_m = a_i$, we can promote $a_i$ to $h_j$ and promote $a_k$ to $h_l$. If $a_m \neq a_i$, we promote $a_i$ to $h_j$, promote $a_k$ to $h_l$ and demote $a_m$ to $l(a_m)$ to obtain a more popular matching than $M$.

– *Case (ii):* $h_j \in f(a_k)$. As $h_j \in s(a_i)$, $h_j$ must be an even vertex by our definition of an $s$-house. Let $G_1$ be the first-choice graph of $I$ as previously defined. Let $M_1 = M \cap E_1$. Then $M_1$ is a maximum matching in $G_1$ by Lemma 4.3.1. Furthermore, there exists an alternating path $P$ of even length to $h_j$ in $G_1$, with respect to $M_1$, from some (even) house $h_l$, which is undersubscribed in $M_1$. Let $M' = M \setminus \{(a_i, M(a_i))\}$. We consider the subcases that (a) $h_l$ is undersubscribed in $M'$ or (b) $h_l$ is full in $M'$. In subcase (a), we can reuse the proof of Case (i) in Lemma 4.3.1 to obtain a matching $M''$ by matching $a_i$ with $h_j$, and then matching all other agents in $P$ with one of their other first-choice houses in $P$ by augmenting along $P$. It follows that $M''$ is more popular than $M$, a contradiction. In subcase (b), we can always find an agent $a_m \in M'(h_l) \setminus f(h_l)$. The remainder of our proof then follows a similar argument to that used in Case (ii) of Lemma 4.3.1 where we can obtain a matching $M''$ that is more popular than $M$, a contradiction. □

As was the case with CHA, we can also define a subgraph $G'$ for the CHAT instance $I$ by this time letting $G'$ contain only edges from each agent $a_i$ to houses in $f(a_i) \cup s(a_i)$. Now, all popular matchings must be agent-complete in $G'$ in view of last resort houses. However, an agent-complete matching need not exist if $s(a_i) \neq \{l(a_i)\}$ for some agent $a_i$. Lemmas 4.3.1, 4.3.5 and 4.3.6 give rise to the following characterisation of popular matchings in $I$.

**Theorem 4.3.1.** *A matching $M$ is popular in $I$ if and only if*

1. *$M \cap E_1$ is a maximum matching in $G_1$, and*

2. *$M$ is an agent-complete matching in the subgraph $G'$.*

*Proof.* By Lemmas 4.3.1, 4.3.5 and 4.3.6, any popular matching necessarily satisfies Conditions 1 and 2. We now show that these conditions are sufficient.

Let $M$ be any matching satisfying Conditions 1 and 2. Suppose for a contradiction that $M'$ is a matching that is more popular than $M$. Let $a_i$ be any agent that prefers $M'$ to $M$. Since $a_i$ prefers $M'(a_i)$ to $M(a_i)$, $M$ is an agent-complete matching in $G'$, and $G'$ only contains edges from $a_i$ to $f(a_i) \cup s(a_i)$, it follows that $M(a_i) \in s(a_i)$, and $f(a_i)$ and

$s(a_i)$ are disjoint. Hence, $M'(a_i)$ must be an odd or unreachable house in $G_1$, as $M(a_i)$ is an even house of highest rank in $a_i$'s preference list.

Let $h_{j_1} = M'(a_i)$. Since $h_{j_1}$ is odd or unreachable, it follows by Condition 1 and Lemma 4.3.4(b) that $|M(h_{j_1})| = c_{j_1}$ and $M(h_{j_1}) \subseteq f(h_{j_1})$. Now since $a_i \in M'(h_{j_1}) \backslash M(h_{j_1})$, there exists a distinct agent $a_{k_1} \in M(h_{j_1}) \backslash M'(h_{j_1})$. If $a_{k_1}$ is unassigned in $M'$ or $M'(a_{k_1}) \notin f(a_{k_1})$, then $a_{k_1}$ prefers $M$ to $M'$. Otherwise, suppose $M'(a_{k_1}) \in f(a_{k_1})$. Let $h_{j_2} = M'(a_{k_1})$. It follows that $a_{k_1}$ is even or unreachable so that $h_{j_2}$ must be odd or unreachable. It then follows by Condition 1 and Lemma 4.3.4(b) that $|M(h_{j_2})| = c_{j_2}$ and $M(h_{j_2}) \subseteq f(h_{j_2})$. Hence, there exists an agent $a_{k_2} \neq a_{k_1}$ such that $a_{k_2} \in M(h_{j_2}) \backslash M'(h_{j_2})$ and $h_{j_2} \in f(a_{k_2})$. If $a_{k_2}$ is unassigned in $M'$ or $M'(a_{k_2}) \notin f(a_{k_2})$, then $a_{k_2}$ prefers $M$ to $M'$. Otherwise, suppose that $M'(a_{k_2}) \in f(a_{k_2})$. Let $h_{j_3} = M'(a_{k_2})$. Then there exists an agent $a_{k_3} \in M(h_{j_3}) \backslash M'(h_{j_3})$ by a similar argument for $a_{k_2}$. Note that possibly $h_{j_3} = h_{j_1}$, but we must be able to choose $a_{k_3} \neq a_{k_1}$, for otherwise $|M'(h_{j_1})| > |M(h_{j_1})|$, which is a contradiction since $|M(h_{j_1})| = c_{j_1}$. Thus, $a_{k_3}$ is a distinct agent, so that we can repeat the above argument to identify an alternating path $P$ in which houses need not be distinct, but agents are distinct. It follows that $P$ must terminate at some agent $a_{k_r}$ as the number of agents are finite. Furthermore, it must be the case that $a_{k_r}$ is unassigned in $M'$ or $M'(a_{k_r}) \notin f(a_{k_r})$ so that for every $a_i$ that prefers $M'$ to $M$, there must exist a distinct $a_{k_r}$ that prefers $M$ to $M'$.

Finally, we note the uniqueness of $a_{k_r}$. If there exists another agent $a_i'$ who prefers $M'$ to $M$, then we can build another alternating path – it is possible that some of the houses are those already used in previous alternating paths such as $P$. However, it must be the case (from our argument that $a_{k_3}$ is a distinct agent) that we are always able to identify distinct agents not already used in previous alternating paths, as each house on the path is odd or unreachable, and thus full in $M$. Hence, $M$ is popular in $I$. □

### 4.3.2 Finding a popular matching

Theorem 4.3.1 leads to Algorithm Popular-CHAT for finding a popular matching in an instance $I$ of CHAT or reporting that none exists, as shown in Algorithm 7. The next lemma is an important step in establishing the correctness of the algorithm.

**Lemma 4.3.7.** *Algorithm Popular-CHAT constructs a matching $M$ such that $M \cap E_1$ is a maximum matching of $G_1$.*

**Algorithm 7** Algorithm Popular-CHAT

1: Build subgraph $G_1$=$(A, H, E_1)$, where $E_1$=$\{(a_i, h_j) : a_i \in A \wedge h_j \in f(a_i)\}$.
2: Compute a maximum matching $M_1$ of first-choice edges in $G_1$.
3: Obtain an EOU labelling of $G_1$.
4: Build subgraph $G'$=$(A, H, E')$, where $E'$=$\{(a_i, h_j) : a_i \in A \wedge h_j \in f(a_i) \cup s(a_i)\}$.
5: Delete all edges in $G'$ connecting two odd vertices, or connecting an odd vertex with an unreachable vertex. (This step does not delete an edge of $M_1$.)
6: Find a maximum matching $M$ in the reduced graph $G'$ by augmenting $M_1$.
7: **if** $M$ is not agent-complete in $G'$ **then**
8: **output** "No popular matching exists";
9: **else**
10: **return** $M$ as a popular matching in $I$;

*Proof.* We firstly claim that $G'$ does not contain any edges of rank greater than 1 incident to odd vertices and unreachable houses. Now, it follows by our definition of $s$-houses, for any odd or unreachable house $h_j \in H$, $h_j \notin s(a_i)$ for any agent $a_i \in A$. Thus, there exist only first-choice edges incident to any such $h_j$. By Lemma 4.3.4(b), every odd agent $a_i$ in $G_1$ can only be assigned in any maximum matching of $G_1$ to some even house $h_k$. Since $(a_i, h_k)$ is a first-choice edge in $G_1$ and $s(a_i)$ defines the highest-ranked even house in $a_i$'s preference list, it follows that $s(a_i) \subseteq f(a_i)$. Hence, the claim is established.

Hence by the above claim, it follows that the edges removed from $G'$ during Step 5 of the algorithm, between two odd vertices or between an odd vertex and an unreachable vertex, are first-choice edges in $G'$. However by Lemma 4.3.4(c), no maximum matching in $G_1$ can contain these edges. Thus, no popular matching can contain these edges by Lemma 4.3.1. In particular, no edge of $M_1$ is deleted by Step 5.

It also follows by Lemma 4.3.4(c) that there cannot exist any (first-choice) edges in $G_1$ between two even vertices, or between an even and an unreachable vertex. As a result, the only first-choice edges that remain in $G'$ after the edge deletions are those edges between (i) odd agents and even houses, (ii) even agents and odd houses, and (iii) unreachable agents and unreachable houses. Define a *second-choice* edge as belonging to the edge set $\{(a_i, h_j) \in E' : h_j \in s(a_i) \wedge s(a_i) \not\subseteq f(a_i)\}$. Then by the above claim, the only second-choice edges that remain in $G'$ are those between even agents and even houses, and between unreachable agents and even houses.

The matching $M$ is obtained from $M_1$ through successive augmentation in Gabow's algorithm. We claim that there does not exist any augmenting path $P$ in which an un-

reachable agent $a_i$ (who is assigned in $M_1$ to some house in $f(a_i)$) becomes worse off, for suppose otherwise. We trace the path $P$ from the undersubscribed house endpoint. Let $a_i$ be the first unreachable agent to become worse off after we augment along $P$. Let $M^b$ be the matching before we augmented along $P$ and let $M^a$ be the matching obtained from satisfying $P$. Assume that $a_i$ is assigned to $h_{j_1}$ in $M^b$. Then, it follows that $h_{j_1}$ is unreachable. Furthermore, we can pick an agent $a_{i_1} \neq a_i$ assigned to $h_{j_1}$ in $M^a$ but not in $M^b$. It follows that $a_{i_1}$ must be unreachable because any unreachable house has only incident edges from unreachable agents in $G'$ and since any unreachable house does not have any edge of rank greater than 1 incident to it as established above. If $a_{i_1}$ is unassigned in $M^b$, then we have finished tracing the path $P$. However, this gives a contradiction by Lemma 4.3.4(b). Hence, $a_{i_1}$ must be assigned to some first-choice house $h_{j_2}$ in $M^b$ or else $a_i$ cannot be the first unreachable agent to become worse off. It thus follows that $h_{j_2}$ must also be unreachable. We can repeat the above argument to trace the path $P$ until we terminate at some agent $a_{i_r}$ who is assigned to the unreachable house $h_{j_r}$ in $M^a$. It is evident that $a_{i_r}$ must be unassigned in $M^b$. However, any such $a_{i_r}$ must be unreachable, which is a contradiction again by Lemma 4.3.4(b).

Now, since all odd agents have only first-choice edges incident to them in $G'$, they must remain assigned to first-choice houses in $M$ even if they participated in any augmenting paths. Moreover, it must be the case that the odd houses, each of which is full in $M_1$, must be full in $M$ and incident only to first-choice edges in $M$ (since odd houses are incident only to such edges in $G'$). Finally, by the above paragraph, unreachable agents cannot become worse off in $M$ than in $M_1$. Hence, only even agents may become worse off in $M$ than in $M_1$, but this means that at least $|\mathcal{O}_A| + \sum_{h_j \in \mathcal{O}_H} c_j + |\mathcal{U}_A|$ first-choice edges assigned previously in $M_1$ remain assigned in $M$. It thus follows by Lemma 4.3.4(b) that $M \cap E_1$ is a maximum matching of $G_1$. □

Hence if Algorithm Popular-CHAT returns a matching $M$, then $M$ is both an agent-complete matching in $G'$ and $M \cap E_1$ is a maximum matching of $G_1$ by Lemma 4.3.7. Hence $M$ is a popular matching in $I$ by Theorem 4.3.1.

We now consider the complexity of Algorithm Popular-CHAT. Let $F$ be the number of first-choice edges in $G$. It is straightforward to see that $G_1$ can be constructed in $O(F + n_2)$ time. We use Gabow's algorithm [15] to compute a maximum matching $M_1$ in $G_1$ in $O(\sqrt{C}F)$ time. We then obtain an EOU labelling of $G_1$ as follows. We first use a pre-processing step to label each unassigned agent and each undersubscribed house as

even. This step takes $O(n)$ time. Next, restricted breadth-first search may be used on $G_1$ to search for alternating paths with respect to $M_1$, building up odd or even labels for every vertex encountered. This step labels all odd and even (assigned) agents, and all odd and even (full) houses and takes $O(m)$ time. Any remaining unlabelled vertices must be unreachable and we can directly label these vertices in $G_1$ in $O(n)$ time. Thus, the total time complexity of this step is $O(n + m)$. The EOU labelling of $G_1$ is then used to construct $G'$ and to delete certain edges from $G'$ at Steps 4 and 5 of the algorithm, both of which take $O(m)$ time overall. Finally, we then use Gabow's algorithm again to obtain the maximum matching $M$ in $G'$ in $O(\sqrt{C}(F + S))$ time, where $S$ is the number of second-choice edges in $G'$. The following result gives the overall run-time of Algorithm Popular-CHAT.

**Lemma 4.3.8.** *Given an instance of* CHAT*, we can find a popular matching, or determine that none exists, in* $O(\sqrt{C}m)$ *time.*

### 4.3.3 Finding a maximum popular matching

It now remains to consider the problem of finding a maximum popular matching in $I$. The aim is to find a matching that satisfies the conditions of Theorem 4.3.1 and that minimises the number of agents who are assigned to their last resort houses. We begin by firstly using Algorithm Popular-CHAT to compute a popular matching $M$ in $I$, assuming such a matching exists. Then $M \cap E_1$ is a maximum matching in $G_1$. We remove all edges in $G'$ (and thus from $M$) that are incident to a last resort house. It follows that $M$ still satisfies the property that $M \cap E_1$ is a maximum matching in $G_1$, but $M$ need not be maximum in $G'$ if agents become unassigned as a result of the edge removals. Thus, we obtain a new maximum matching $M'$ from $M$ by using Gabow's algorithm on $G'$ again. If $M'$ is not agent-complete in $G'$, we simply assign any agent who remains unassigned in $M'$ to their last resort house to obtain an agent-complete matching. Using an argument similar to that in the proof of Lemma 4.3.7, it follows that $M' \cap E_1$ is a maximum matching of $G_1$. Thus, $M'$ is a maximum popular matching in $I$. Now, it is straightforward to see that the overall complexity of this approach is as for Algorithm Popular-CHAT, giving the following result.

**Theorem 4.3.2.** *Given an instance of* CHAT*, we can find a maximum popular matching, or report that no such matching exists, in* $O(\sqrt{C}m)$ *time.*

### 4.3.4 "Cloning versus our direct approach

We may compare the complexity of our direct approach for CHAT to that obtained using cloning on $I$ together with the algorithm of [4] on the cloned instance of $I$. As in Section 4.2, the latter approach takes $\Omega(\sqrt{n_1 + C}m')$ time, where $m' = \sum_{a_i \in A} \sum_{h_j \in A_i} c_j$. It follows that this is slower than our direct algorithm by a factor of $\Omega(m'/m)$.

## 4.4 Open problem

We conclude this chapter with the following open problem.

Let $I$ be an extension of CHA in which each agent now has a capacity to be assigned to more than one house simultaneously in any matching $M$ of $I$, i.e. a many-many mapping is sought in any matching of agents to houses in $I$. We remark that it may be appropriate to redefine an agent's preference over matchings to Definition 1.4.1 in this setting. However, the definition of a popular matching is unchanged. Is the problem of finding a popular matching (or reporting that none exists) then solvable in polynomial time? It is not immediately clear if cloning offers a straightforward solution in this context, since both agents and houses have capacities, so that it would be hard to avoid assigning the same agent to the same house more than once in any cloning approach. Hence, a direct algorithm using an approach of the kind in this chapter is likely to be required. This then raises the question: if a polynomial-time algorithm exists for solving this problem in CHA, can we extend this to solve the analogous problem in CHAT?

# Chapter 5

# Popular matchings in WCHA

## 5.1 Introduction

In this chapter, we extend our study of popular matchings from Chapter 4 to the Weighted Capacitated House Allocation problem with no ties (WCHA), a generalisation of CHA in which each agent has an associated weight that indicates his priority. The assignment of weights to agents allow us to build up a spectrum of priority levels for agents in the competition for houses in situations where the total capacity of the houses is less than the number of agents. In turn, this gives some agents a better chance of "doing well". For instance, the assignment of weights can enable DVD rental companies like Amazon to give priority to those members who have paid more for privileged status whenever a certain title is limited in stock. Alternatively, weights may be assigned to candidates in job markets based on objective criteria such as academic results or relevant work experience.

The main results of this chapter, and their organisation are as follows. In Section 5.2, we first develop necessary conditions for a matching to be popular in a WCHA instance $I$. In Section 5.3, we identify a structure in the underlying graph of $I$ that singles out those edges that cannot belong to a popular matching. We then use these two results in conjunction to construct a $O(\sqrt{C}n_1 + m)$ time algorithm for finding a maximum popular matching in $I$ or reporting that none exists, where $C$ is the total capacity of the houses, $n_1$ is the number of agents and $m$ is the total length of preference lists in $I$. Finally, as for the case of CHA, a straightforward solution to the problem of finding a maximum popular matching in $I$ may be to use "cloning". Informally, this entails creating $c_j$ clones for each house $h_j$ to obtain an instance $J$ of WHAT, and then applying the algorithm of [43] to $J$. However, we will show in Section 5.3.6 that this approach leads to a slower algorithm

than the direct approach that we will be using, as was the case for CHA.

## 5.2 Characterising a popular matching

We remark that all the notations and terminology that were defined for CHA in Chapters 1, 2 and 4 carry over directly to WCHA. We also provide some additional definitions and redefine certain concepts that we need to adapt to WCHA as follows.

Let $I$ be an instance of WCHA, and let $G = (A, H, E)$ be the underlying graph of $I$, where $A = \{a_1, a_2, ..., a_{n_1}\}$ is the set of *agents*, $H = \{h_1, h_2, ..., h_{n_2}\}$ is the set of *houses* and $E$ is the set of edges in $G$ representing the acceptable houses of the agents. As was the case in CHA, we also create a unique *last resort* house $l(a)$ for each agent $a$ and append $l(a)$ to $a$'s preference list. We henceforth assume that $G$ contains the vertex $l(a)$ and the edge $(a, l(a))$ for each $a \in A$. Again, we let $n = n_1 + n_2$ and $m = |E|$.

Every agent $a$ has a positive weight $w(a)$ indicating the priority of the agent, and we partition $A$ into the sets $P_1$, $P_2$, ..., $P_k$, such that the weight of agents in $P_z$ is $w_z$, and $w_1 > w_2 > ... > w_k > 0$. For each agent $a \in A$, we say that $a$ has *priority* $z$ if $a \in P_z$, and we use $P(a)$ to denote the priority of $a$, that is $P(a) = z$. We assume that no agent has an empty preference list and each house is acceptable to at least one agent, i.e. $m \geq \max\{n_1, n_2\}$. We also assume that $c_j \leq n_1$ for each $h_j \in H$. Again, let $C = \sum_{j=1}^{n_2} c_j$ denote the sum of the capacities of the houses.

As with CHA, given two matchings $M$ and $M'$ in $I$, we say that an agent $a$ *prefers* $M'$ to $M$ if either (i) $a$ is matched in $M'$ and unmatched in $M$, or (ii) $a$ is matched in both $M'$ and $M$ and prefers $M'(a)$ to $M(a)$. Let $P(M', M)$ denote the set of agents who prefer $M'$ to $M$. Then, in view of the weights assigned to agents, it is appropriate to define a popular matching in the context of WCHA as follows. Firstly, let the *satisfaction* of $M'$ with respect to $M$ be defined as $sat(M', M) = \sum_{a \in P(M', M)} w(a) - \sum_{a \in P(M, M')} w(a)$. We then say that $M'$ is *more popular than* $M$ if $sat(M', M) > 0$. A matching $M$ in $I$ is *popular* if there is no other matching in $I$ that is more popular than $M$.

Let us now proceed to obtain a characterisation of popular matchings in $I$. For each agent $a \in A$, we introduce the notion of $a$'s *$f$-house* and $a$'s *$s$-house* denoting these by $f(a)$ and $s(a)$ respectively. Intuitively, $f(a)$ is the most preferred house on $a$'s preference list to which $a$ could be assigned in a popular matching. We use Algorithm Label-f shown in Algorithm 8 to define $f(a)$ precisely.

**Algorithm 8** Algorithm Label-f

1: **for each** $h_j \in H$ **do**
2: **for** $i$ in $1..k$ **do**
3: $f_{i,j} := 0$;
4: **for each** $a \in P_1$ **do**
5: $f(a) :=$ first-ranked house $h_j$ on $a$'s preference list;
6: $f_{1,j}$ ++;
7: **for** $z$ in $2..k$ **do**
8: **for each** $a \in P_z$ **do**
9: $q := 1$;
10: $h_j :=$ house at position $q$ on $a$'s preference list;
11: **while** $(\sum_{p=1}^{z-1} f_{p,j} \geq c_j)$ **do**
12: $q$ ++;
13: $h_j :=$ house at position $q$ on $a$'s preference list;
14: $f(a) := h_j$;
15: $f_{z,j}$ ++;

Here, we will define the $f$-houses for all the agents in phases, with each phase corresponding to a priority level $P_z$. Intuitively, during the course of the algorithm's execution, $f_{i,j}$ will denote the number of agents with priority $i$ whose $f$-house is defined and equal to $h_j$. Initially, $f_{i,j} = 0$ for all $i$ $(1 \leq i \leq k)$ and $j$ $(1 \leq j \leq n_2)$. We then define the $f$-house for each agent as follows. For every agent $a \in P_1$, we let $f(a)$ be the first-ranked house $h_j$ on $a$'s preference list, and we call such a house an $f_1$-*house*. Given $2 \leq z \leq k$, for every agent $a \in P_z$, we let $f(a)$ be the most-preferred house $h_j$ on $a$'s preference list such that $\sum_{p=1}^{z-1} f_{p,j} < c_j$ – we call $h_j$ an $f_z$-*house*. Clearly, the algorithm must terminate due to the presence of a unique last resort house at the end of each agent's preference list. Once the algorithm has terminated, we let $f_i(h_j)$ denote the set $\{a \in P_i : f(a) = h_j\}$. Then, $f_{i,j} = |f_i(h_j)|$ (possibly $f_{i,j} = 0$). Here, and henceforth throughout this chapter, any reference to $f_{i,j}$ refers to the value of this variable upon termination of Algorithm Label-f.

It is straightforward to verify that Algorithm Label-f runs in $O(m)$ time if we use virtual initialisation (described in [9, p.149]) for the steps in lines 1-3. The example in Figure 5.1 gives an illustration of the definition of $f$-houses. Here, the $f$-houses of the agents are as follows: $f(a_1) = h_1$, $f(a_2) = h_3$, $f(a_3) = h_3$, $f(a_4) = h_4$, $f(a_5) = h_4$ and $f(a_6) = h_4$.

| Agent | Priority | Weight | Pref list | | | | House | Capacity |
|---|---|---|---|---|---|---|---|---|
| $a_1$: | 1 | 7 | $h_1$ | $h_2$ | $h_3$ | | $h_1$ | 1 |
| $a_2$: | 2 | 4 | $h_1$ | $h_3$ | $h_4$ | | $h_2$ | 2 |
| $a_3$: | 2 | 4 | $h_3$ | $h_5$ | | | $h_3$ | 2 |
| $a_4$: | 3 | 2 | $h_3$ | $h_1$ | $h_4$ | $h_5$ | $h_4$ | 2 |
| $a_5$: | 3 | 2 | $h_1$ | $h_4$ | $h_5$ | | $h_5$ | 1 |
| $a_6$: | 3 | 2 | $h_4$ | $h_1$ | $h_2$ | | | |

Figure 5.1: An instance $I_1$ of WCHA

Now, for each $h_j \in H$, let $f(h_j) = \{a \in A : f(a) = h_j\}$ and $f_j = |f(h_j)|$ (possibly $f_j = 0$), i.e. $f(h_j) = \bigcup_{p=1}^{k} f_p(h_j)$. Clearly each $h_j$ may be an $f_z$-house for more than one priority level $z$. For every such $h_j$, let us define $d_j$ to denote the priority level such that

$$d_j = \begin{cases} \max\{r : 0 \leq r \leq k \wedge f_{r,j} \neq 0\}, \text{if } f_j \leq c_j, \\ \max\{r : 0 \leq r \leq k \wedge \sum_{i=1}^{r} f_{i,j} < c_j\}, \text{if } f_j > c_j. \end{cases}$$

Note that for every $h_j$ such that $f_j > c_j$, clearly $\bigcup_{p=d_j+1}^{k} f_p(h_j) \neq \emptyset$. Hence, for every such $h_j$, we define $g_j$ to be the priority level such that $g_j = \max\{r : d_j < r \leq k \wedge f_{r,j} \neq 0\}$. We refer to Figure 5.1 for illustration. Here, $d_1 = 1$, $d_3 = 2$ and $d_4 = 2$. Note that $d_2$ and $d_5$ are not defined, for $h_2$ and $h_5$ are not $f$-houses for any agent. Also, since $f_4 > c_4$, it follows that $g_4 = 3$; however, $h_4$ is the only $f$-house $h_j$ such that $f_j > c_j$. We now work towards obtaining a characterisation of popular matchings in WCHA. We begin with the following technical lemma.

**Lemma 5.2.1.** *Let $M$ be a matching in any* WCHA *instance $I$. Let $h_j \in H$ be a house, and let $1 \leq i \leq d_j$. Let $a \in A$ be an agent such that $a \in P_i$ and $a \in f_i(h_j) \backslash M(h_j)$. If $h_j$ is full in $M$ and $\bigcup_{p=1}^{i-1} f_p(h_j) \subseteq M(h_j)$, then there exists some agent in $M(h_j) \backslash \bigcup_{p=1}^{i} f_p(h_j)$.*

*Proof.* Let $F = \bigcup_{p=1}^{i} f_p(h_j)$. Then, it follows that $F = (M(h_j) \cap F) \cup (F \backslash M(h_j))$ and $M(h_j) = (M(h_j) \cap F) \cup (M(h_j) \backslash F)$. Hence, we have that $|F| = \sum_{p=1}^{i} f_{p,j} = |M(h_j) \cap F| + |F \backslash M(h_j)|$ and $|M(h_j)| = |M(h_j) \cap F| + |M(h_j) \backslash F|$. Clearly, $|F| \leq \sum_{p=1}^{d_j} f_{p,j} \leq c_j$. Since $a \in f_i(h_j) \backslash M(h_j)$, it follows that $|F \backslash M(h_j)| > 0$. Hence, $|M(h_j) \backslash F| = |M(h_j)| - |F| + |F \backslash M(h_j)| > 0$. □

The next three lemmas contribute to the characterisation of popular matchings in WCHA.

**Lemma 5.2.2.** *Let $M$ be a popular matching in any given* WCHA *instance $I$ and let $a \in A$ be any agent. Then, $a$ cannot be assigned to a house better than $f(a)$ in $M$.*

*Proof.* Let $a$ be the agent with lowest priority (i.e. greatest weight) such that $a$ is assigned to house $h_j$ in $M$ and suppose that $a$ prefers $h_j$ to $f(a) = h_l$. Let $a \in P_i$ so that $\sum_{p=1}^{i-1} f_{p,j} \geq c_j$ by definition of $f(a)$ as $a$'s $f$-house. Clearly, there must be no agent $a'$ such that $a' \in P_z$ where $z \geq i$ and $f(a') = h_j$, for otherwise $\sum_{p=1}^{z-1} f_{p,j} < c_j$, a contradiction. Let $a'$ be any agent with priority level $z < i$ such that $a' \in f(h_j) \backslash M(h_j)$ – there must exist such an agent since $\bigcup_{p=i}^{k} f_p(h_j) = \emptyset$ and $\sum_{p=1}^{i-1} f_{p,j} \geq c_j$ and $a \in M(h_j)$. Then, by choice of $a$, $a'$ is assigned in $M$ to a house worse than $f(a')$. However, this means that we can promote $a'$ to $f(a')$ and demote $a$ to $l(a)$ to obtain a matching whose improvement in satisfaction is $w_z - w_i > 0$, a contradiction. □

**Lemma 5.2.3.** *Let $M$ be a popular matching in any given* WCHA *instance $I$. Then, for each $h_j \in H$, $\bigcup_{i=1}^{d_j} f_i(h_j) \subseteq M(h_j)$.*

*Proof.* Given $1 \leq i \leq d_j$, we will prove by induction on $i$ that $f_i(h_j) \subseteq M(h_j)$.

For the base case, let $i = 1$. Suppose that $f_1(h_j) \not\subseteq M(h_j)$. Then, there exists some agent $a_r \in f_1(h_j) \backslash M(h_j)$. By definition of an $f_1$-house, $h_j$ must be the first house on $a_r$'s preference list. Hence, $a_r$ prefers to be assigned to $h_j$ than $M(a_r)$. Clearly, if $h_j$ is undersubscribed in $M$, we can promote $a_r$ to $h_j$ to obtain a matching more popular than $M$, a contradiction. Hence, $h_j$ is full in $M$. Choose any $a_s \in M(h_j) \backslash f_1(h_j)$ (which must exist by Lemma 5.2.1). Since $a_s \notin f_1(h_j)$, either (i) $a_s$ has priority $> 1$, or (ii) $a_s$ has priority 1 but $f(a_s) = h_l \neq h_j$. In subcase (i), we can promote $a_r$ to $h_j$ and demote $a_s$ to $l(a_s)$ to obtain a more popular matching. In subcase (ii), since $f(a_s) = h_l$, it follows by Lemma 5.2.2 that $a_s$ prefers to be assigned to $h_l$ than $h_j$. Now, if $h_l$ is undersubscribed in $M$, we can promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching. Hence, $h_l$ is full in $M$. If $h_l = M(a_r)$, then we can then promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching. Otherwise, choose any $a_t \in M(h_l)$. Clearly, $a_t \neq a_r$. We can then promote $a_r$ to $h_j$, promote $a_s$ to $h_l$, and demote $a_t$ to $l(a_t)$ to obtain a matching whose improvement in satisfaction is $w_1 + w_1 - w(a_t) > 0$.

For the inductive case, assume that $2 \leq i \leq d_j$, and if $q < i$, then $f_q(h_j) \subseteq M(h_j)$ for all $h_j \in H$. Suppose for a contradiction that $f_i(h_j) \not\subseteq M(h_j)$. Then, there exists some $a_r \in f_i(h_j) \backslash M(h_j)$. Now, since $f(a_r) = h_j$, it follows by Lemma 5.2.2 that $a_r$ must prefer to be assigned to $h_j$ than $M(a_r)$. Thus, if $h_j$ is undersubscribed in $M$, we can

promote $a_r$ to $h_j$ to obtain a more popular matching than $M$, a contradiction. Hence, $h_j$ is full in $M$. Choose any $a_s \in M(h_j) \setminus \bigcup_{p=1}^{i} f_p(h_j)$ which must exist by Lemma 5.2.1. Since $a_s \notin \bigcup_{p=1}^{i} f_p(h_j)$, either (i) $a_s$ has priority $> i$, or (ii) $a_s$ has priority $\leq i$ but $f(a_s) = h_l \neq h_j$.

In subcase (i), we can promote $a_r$ to $h_j$ and demote $a_s$ to $l(a_s)$ to obtain a more popular matching than $M$, a contradiction. In subcase (ii), suppose that $a_s$ has priority $z < i$. Then $h_l$ is an $f_z$-house so that $a_s \in f_z(h_l)$. However, this is a contradiction since by the inductive hypothesis $f_z(h_l) \subseteq M(h_l)$, but $M(a_s) \neq h_l$. Thus, $a_s$ has priority $i$ and $a_s \in f_i(h_l)$. Clearly, since $f(a_s) = h_l$, it follows by Lemma 5.2.2 that $a_s$ must prefer to be assigned to $h_l$ than $h_j$. Thus, if $h_l$ is undersubscribed, we can promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching than $M$, a contradiction. Hence $h_l$ is full. If $h_l = M(a_r)$, then we can promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching. Otherwise, $h_l \neq M(a_r)$. We will show how to choose $a_t \in M(h_l)$. Since $f(a_s) = h_l$ and $2 \leq i \leq k$, by our definition of $f$-houses, $h_l$ must be the most preferred house on $a_s$'s preference list such that $\sum_{p=1}^{i-1} f_{p,l} < c_l$.

Now, by the inductive hypothesis, it must be the case that $\bigcup_{p=1}^{i-1} f_p(h_l) \subseteq M(h_l)$. Since $\sum_{p=1}^{i-1} f_{p,l} < c_l$ and $h_l$ is full, it follows that $\bigcup_{p=1}^{i-1} f_p(h_l) \subset M(h_l)$. Hence, it must be the case that $M(h_l) \setminus \bigcup_{p=1}^{i-1} f_p(h_l) \neq \emptyset$. It follows that there exists some agent $a_t \in M(h_l) \setminus \bigcup_{p=1}^{i-1} f_p(h_l)$ and, either (i) $a_t \in \bigcup_{p=i}^{k} f_p(h_l)$ or (ii) $a_t \notin f(h_l)$. Clearly, in case (ii), $a_t$ has priority $\geq i$ by a similar argument for $a_s$. For, if $a_t$ has priority $z < i$, then by the inductive hypothesis, since $h_m = f(a_t)$ is an $f_z$-house and $a_t \in f_z(h_m)$, it follows that $f_z(h_m) \subseteq M(h_m)$. However, this gives a contradiction since $M(a_t) \neq h_m$. Hence, $a_t$ has priority $\geq i$ in both cases (i) and (ii). We can then promote $a_r$ to $h_j$, promote $a_s$ to $h_l$ and demote $a_t$ to $l(a_t)$ to obtain a matching whose improvement in satisfaction is $w_i + w_i - w(a_t) > 0$, a contradiction. □

**Lemma 5.2.4.** *Let $M$ be a popular matching in any given* WCHA *instance $I$. Then, for each $h_j \in H$, if $f_j > c_j$, then $M(h_j) \setminus \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$.*

*Proof.* Clearly, $f_{g_j,j} > c_j - \sum_{p=1}^{d_j} f_{p,j}$. It follows by Lemma 5.2.3 that $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ so that no matter whether $h_j$ is full or undersubscribed, $f_{g_j}(h_j) \not\subseteq M(h_j)$. Hence, there exists some agent $a_r$ such that $a_r \in f_{g_j}(h_j) \setminus M(h_j)$. Note that $a_r$ has priority $g_j$. Clearly, since $f(a_r) = h_j$, $a_r$ must prefer to be assigned to $h_j$ than $M(a_r)$ by Lemma 5.2.2. Hence, if $h_j$ is undersubscribed, we can promote $a_r$ to $h_j$ to obtain a more popular matching than $M$,

a contradiction. It follows that $h_j$ is full. We will show that $M(h_j)\setminus\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$ for all values of $d_j$.

If $d_j = 0$, then it must be the case that $f_{1,j} > c_j$ and $a_r \in f_1(h_j)\setminus M(h_j)$. If $M(h_j) \subseteq f_1(h_j)$, then the result is immediate. Hence, suppose that $M(h_j) \not\subseteq f_1(h_j)$. Choose any $a_s \in M(h_j)\setminus f_1(h_j)$. Clearly, either (i) $a_s$ has priority 1 but $f(a_s) = h_l \neq h_j$ or (ii) $a_s$ has priority $> 1$. In case (i), since $f(a_s) = h_l$, $a_s$ must prefer to be assigned to $h_l$ than $h_j$ by Lemma 5.2.2. Hence, if $h_l$ is undersubscribed, we can promote $a_r$ to $h_j$ and $a_s$ to $h_l$ to obtain a more popular matching, a contradiction. Thus, $h_l$ is full. By Lemma 5.2.3, $\bigcup_{p=1}^{d_l} f_p(h_l) \subseteq M(h_l)$. Since $a_s \in f_1(h_l)\setminus M(h_l)$, it follows that $d_l = 0$, i.e. $f_{1,l} > c_l$. Now, if $M(a_r) = h_l$, then we can promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching. Hence, $M(a_r) \neq h_l$. Choose any $a_t \in M(h_l)$. We then promote $a_r$ to $h_j$, promote $a_s$ to $h_l$ and demote $a_t$ to $l(a_t)$ to obtain a matching whose improvement in satisfaction is $w_1 + w_1 - w(a_t) > 0$. In case (ii), we can promote $a_r$ to $h_j$ and demote $a_s$ to $l(a_s)$ to obtain a more popular matching.

Hence, $d_j \geq 1$. Suppose for a contradiction that $M(h_j)\setminus\bigcup_{p=1}^{d_j} f_p(h_j) \not\subseteq f_{g_j}(h_j)$. It follows that there exists some agent $a_s \in M(h_j)\setminus\bigcup_{p=1}^{g_j} f_p(h_j)$. Recall that $a_r$ has priority $g_j$. Clearly, either (i) $a_s$ has priority $\leq g_j$ but $f(a_s) = h_l \neq h_j$, or (ii) $a_s$ has priority $> g_j$. It is immediate in case (ii) that we can promote $a_r$ to $h_j$ and demote $a_s$ to $l(a_s)$ to obtain a more popular matching, a contradiction. Hence, case (i) applies. It follows by Lemma 5.2.2 that $a_s$ prefers to be assigned to $h_l$ than $h_j$, and so, if $h_l$ is undersubscribed, we can then obtain a more popular matching by promoting $a_r$ to $h_j$ and promoting $a_s$ to $h_l$. Hence $h_l$ is full. Now, if $M(a_r) = h_l$, we can then promote $a_r$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching. Hence, $M(a_r) \neq h_l$.

Let $a_s$ have priority $z_1$ so that $z_1 \leq g_j$. By our definition of $f$-houses, since $h_l = f(a_s)$, if $z_1 = 1$, then $h_l$ is the first house on $a_s$'s preference list. Since $h_l$ is full, then choose any $a_t \in M(h_l)$ and let $a_t$ have priority $z_2$. We obtain an improvement in satisfaction of $w(a_r) + w(a_s) - w(a_t) = w_{g_j} + w_1 - w_{z_2} > 0$ by promoting $a_r$ to $h_j$, promoting $a_s$ to $h_l$ and demoting $a_t$ to $l(a_t)$. Hence, it follows that $z_1 > 1$. Then, $h_l$ must be the most preferred house on $a_s$'s preference list such that $\sum_{p=1}^{z_1-1} f_{p,l} < c_l$. By definition of $f(a_s) = h_l$, $z_1 \leq g_l$. Now, by Lemma 5.2.3, $\bigcup_{p=1}^{d_l} f_p(h_l) \subseteq M(h_l)$. However, $a_s \notin M(h_l)$. Hence, it follows that $z_1 > d_l$, i.e. $z_1 = g_l$. Since $\sum_{p=1}^{z_1-1} f_{p,l} < c_l$ and $h_l$ is full, it follows that $\bigcup_{p=1}^{z_1-1} f_p(h_l) \subset M(h_l)$. Hence, we have that $M(h_l)\setminus\bigcup_{p=1}^{z_1-1} f_p(h_l) \neq \emptyset$. It follows that there exists some agent $a_t \in M(h_l)\setminus\bigcup_{p=1}^{z_1-1} f_p(h_l)$. Clearly, either (i) $a_t \in \bigcup_{p=z_1}^{k} f_p(h_l)$ or

(ii) $a_t \notin f(h_l)$.

Note that since $M(a_r) \neq h_l$, $a_t \neq a_r$. Now, in both case (i) and (ii), if $a_t$ has priority $z_2 \geq z_1$, we can then promote $a_r$ to $h_j$, promote $a_s$ to $h_l$ and demote $a_t$ to $l(a_t)$ to obtain a matching whose improvement in satisfaction is $w(a_r)+w(a_s)-w(a_t) = w_{g_j}+w_{z_1}-w(a_t) > 0$, a contradiction. Hence $z_2 < z_1$, and so only case (ii) applies. Let $h_m = f(a_t)$. It is obvious, by Lemma 5.2.2, that $a_t$ prefers to be assigned to $h_m$ than $h_l$. Furthermore, $h_m \neq h_j$, for suppose not. As $z_2 < z_1 \leq g_j$ and $f(a_t)$ is defined, it follows that $z_2 \leq d_j$. By Lemma 5.2.3, $\bigcup_{p=1}^{z_2} f_p(h_j) \subseteq \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ so that $a_t \in M(h_j)$. However, this gives a contradiction since $a_t \in M(h_l)$ and $h_j \neq h_l$. Clearly also, $h_m \neq M(a_r)$ for otherwise, we can promote $a_r$ to $h_j$, promote $a_s$ to $h_l$ and promote $a_t$ to $h_m$ to obtain a more popular matching, a contradiction. Hence, the houses $h_m$, $h_l$, $h_j$ and $M(a_r)$ are distinct. Clearly too, the agents $a_r$, $a_s$ and $a_t$ are distinct for $z_2 < z_1 \leq g_j$ and $a_r \neq a_s$.

We assume that $h_m$ is full, for otherwise we can obtain a contradiction by promoting $a_r$ to $h_j$, promoting $a_s$ to $h_l$ and promoting $a_t$ to $h_m$. Let $a_u \in M(h_m)$. If $z_2 = 1$, then we can promote $a_r$ to $h_j$, promote $a_s$ to $h_l$, promote $a_t$ to $h_m$ and demote $a_u$ to $l(a_u)$ to obtain a new matching with improvement in satisfaction $w(a_r) + w(a_s) + w(a_t) - w(a_u) = w_{g_j} + w_{z_1} + w_1 - w(a_u) > 0$. Hence, $z_2 > 1$. If we let $a_t$ and $a_u$ take the roles of $a_s$ and $a_t$ respectively, then it follows by the argument that we use to define $a_t$ that we are able to choose $a_u$ such that $a_u$ has priority $< z_2$ and $a_u \notin f(h_m)$. It follows that $a_u$ is an agent distinct from $a_r$, $a_s$ and $a_t$ since $P(a_u) < z_2$.

By continuing this argument, it follows that we obtain a sequence of distinct agents $a_0, a_1, a_2, a_3, ...$ where $a_0 = a_r$, $a_1 = a_s$, $a_2 = a_t$, and $a_3 = a_u$. For $i \geq 4$, the above construction indicates that $P(a_i) < P(a_{i-1})$. If this sequence does not terminate as a result of arriving at a contradiction due to any of the above cases, then we are bound to ultimately generate an agent $a_x$ such that $P(a_x) < 1$, which is impossible. □

Lemmas 5.2.3 and 5.2.4 give rise to the following corollary concerning the relation of $f$-houses to popular matchings.

**Corollary 5.2.1.** *Let $M$ be a popular matching in any* WCHA *instance $I$. Then, for every $f$-house $h_j$,*

1. *if $f_j \leq c_j$, then $f(h_j) \subseteq M(h_j)$;*
2. *if $f_j > c_j$, then $|M(h_j)| = c_j$, $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$, and $M(h_j) \setminus \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$.*

*Proof.* In Case 1, if $f_j \leq c_j$, it follows by definition of $d_j$ that $\bigcup_{p=d_j+1}^{k} f_p(h_j) = \emptyset$. Clearly then, $f(h_j) = \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ by Lemma 5.2.3. In Case 2, it follows by Lemmas 5.2.3 and 5.2.4 that $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$, $M(h_j) \setminus \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$ and $|M(h_j)| = c_j$. □

We now define the concept of an $s$-house for each agent. Given a popular matching $M$, if $M(a) \neq f(a)$, then as we shall show, $M(a) = s(a)$. Given $1 \leq z \leq k$, for every agent $a \in P_z$, we define $s(a)$ to be the most preferred house $h_j$ on $a$'s preference list such that $h_j \neq f(a)$ and $\sum_{i=1}^{z} f_{i,j} < c_j$. Note that $s(a)$ may not exist if $f(a) = l(a)$. However, all such agents will be assigned to their $f$-houses in any matching since last resort houses are unique to individual agents.

To illustrate the $s$-house definition, let us look at Instance $I_1$ in Figure 5.1 again. We may verify from the definition of $s$-houses that $s(a_1) = h_2$, $s(a_2) = h_4$, $s(a_3) = h_5$, $s(a_4) = h_5$, $s(a_5) = h_5$ and $s(a_6) = h_2$. Clearly, the set of $f_i$-houses need not be disjoint from the set of $s_j$-houses for $i \neq j$ as seen from this example. Now, since the process of defining $s$-houses is analogous to the algorithm for defining $f$-houses, the time complexity for defining $s$-houses is also $O(m)$.

Now, it may be shown that a popular matching $M$ will only assign an agent $a$ to either $f(a)$ or $s(a)$ as indicated by the next lemma.

**Lemma 5.2.5.** *Let $M$ be a popular matching in any* WCHA *instance $I$. Then, every agent $a \in A$ is assigned in $M$ to either $f(a)$ or $s(a)$.*

*Proof.* Let $a \in P_i$ and let $M(a) = h_x$. Suppose that the statement of this lemma is false. By Lemma 5.2.2, $a$ cannot be assigned to a house better than $f(a)$. Then, besides $f(a)$ or $s(a)$, $h_x$ can either be (i) a house between $f(a)$ and $s(a)$ or (ii) a house worse than $s(a)$.

In case (i), it follows that $h_x$ is an $f$-house such that $\sum_{p=1}^{i} f_{p,x} \geq c_x$, for otherwise $s(a) = h_x$. Hence, $f_x \geq c_x$ and $M(h_x) \subseteq f(h_x)$ by Corollary 5.2.1. However, $a \in M(h_x) \setminus f(h_x)$, a contradiction.

In case (ii), let $h_j = s(a)$. It follows that $a$ must prefer to be assigned to $h_j$ than $M(a) = h_x$. Clearly, $h_j$ is full, for otherwise we can promote $a$ to $h_j$, a contradiction. It follows by our definition of $s$-houses that $\sum_{p=1}^{i} f_{p,j} < c_j$. Hence, by our definition of $d_j$, $i \leq d_j$. Since $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ (by Lemma 5.2.3) and $h_j$ is full, it follows that $\bigcup_{p=1}^{i} f_p(h_j) \subset M(h_j)$ so that $M(h_j) \setminus \bigcup_{p=1}^{i} f_p(h_j) \neq \emptyset$. Hence, there exists some $a_s \in M(h_j) \setminus \bigcup_{p=1}^{i} f_p(h_j)$. It is obvious that either (i) $a_s \in \bigcup_{p=i+1}^{k} f_p(h_j)$, or (ii) $a_s \notin f(h_j)$.

Clearly in case (i), $a_s$ has priority $> i$, so we can promote $a$ to $h_j$ and demote $a_s$ to

$l(a_s)$ to obtain a matching whose improvement in satisfaction is $w_i - w(a_s) > 0$. In case (ii), let $a_s$ have priority $z_1$. It follows that $z_1 \leq i$, for otherwise, we can promote $a$ to $h_j$ and demote $a_s$ to $l(a_s)$ to obtain a new matching whose improvement in satisfaction is $w_i - w_{z_1} > 0$. Let $f(a_s) = h_l$. Clearly, $a_s$ must prefer to be assigned to $h_l$ than $h_j$ by Lemma 5.2.2. If $h_l$ is undersubscribed, we can then promote $a$ to $h_j$ and promote $a_s$ to $h_l$ to obtain a more popular matching, a contradiction. Hence, suppose that $h_l$ is full. Let $a_t \in M(h_l)$.

If $z_1 = 1$, then we can promote $a$ to $h_j$, promote $a_s$ to $h_l$ and demote $a_t$ to $l(a_t)$ to obtain a matching with improvement in satisfaction $w(a) + w(a_s) - w(a_t) = w_i + w_1 - w(a_t) > 0$. Hence, suppose that $z_1 > 1$. Clearly, $h_x \neq h_l$ for suppose otherwise. By Corollary 5.2.1, $h_l$ must be an $f$-house such that $f_l > c_l$ by existence of $a_s$, for otherwise $a_s \in M(h_l)$. It follows that $M(h_l) \subseteq f(h_l)$. Now, if $h_l = h_x$, then this gives us a contradiction since $a \in M(h_l)$ but $h_x \neq f(a)$ for $a$ prefers $s(a)$ to $h_x$.

Hence, $h_l \neq h_x$. Then, $a_t \neq a$. It follows that we can reuse arguments from the proof of Lemma 5.2.4 to obtain a sequence of distinct agents $a_0, a_1, a_2, ...$ where $a_0 = a$, $a_1 = a_s$, and $a_2 = a_t$. For $j \geq 3$, the construction of the sequence indicates that $P(a_i) < P(a_{i-1})$. If this sequence does not terminate as a result of arriving at a contradiction due to any of the cases outlined in Lemma 5.2.4, then we are bound to ultimately generate an agent $a_x$ such that $P(a_x) < 1$, which is impossible. □

Corollary 5.2.1 and Lemma 5.2.5 give rise to the following result.

**Theorem 5.2.1.** *Let $M$ be a popular matching in any given* WCHA *instance $I$.*

1. *For every $f$-house $h_j$,*

   (a) *if $f_j \leq c_j$, then $f(h_j) \subseteq M(h_j)$;*

   (b) *if $f_j > c_j$, then $|M(h_j)| = c_j$, $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$, and $M(h_j) \setminus \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$.*

2. *Every agent $a$ is assigned to either $f(a)$ or $s(a)$.*

## 5.3 Algorithm for finding a popular matching

Let us form a subgraph $G'$ of $G$ by letting $G'$ contain only two edges for each agent $a \in A$, that is, one to $f(a)$ and the other to $s(a)$. It follows that all popular matchings must

be contained in $G'$ by Theorem 5.2.1. However, Theorem 5.2.1 only gives us necessary conditions for a matching to be popular in an instance of WCHA, since not all matchings in $G'$ satisfying these conditions are popular. For, let us consider the example WCHA instance in Figure 5.1. We have at least two matchings which satisfy Conditions 1 and 2 of Theorem 5.2.1: $M_1 = \{(a_1, h_1), (a_2, h_3), (a_3, h_3), (a_4, h_5), (a_5, h_4), (a_6, h_4)\}$ and $M_2 = \{(a_1, h_1), (a_2, h_3), (a_3, h_3), (a_4, h_4), (a_5, h_5), (a_6, h_4)\}$. However, while $M_1$ may be verified to be a popular matching, $M_2$ is not popular because there exists another matching $M_3 = \{(a_2, h_1), (a_3, h_3), (a_4, h_3), (a_5, h_4), (a_6, h_4)\}$ which gives an improvement in satisfaction of $w(a_2) + w(a_4) + w(a_5) - w(a_1) = 4 + 2 + 2 - 7 > 0$ over $M_2$. Hence, we will "enforce" the sufficiency of the conditions by removing certain edges in $G'$ that cannot form part of any popular matching in $I$. We show how to do this by first introducing the notion of a *potential improvement path* or *PIP* in short, which generalises the concept of a *promotion path* from [43] to WCHA.

### 5.3.1 Potential improvement paths

Let us now define a matching $M$ that satisfies Conditions 1 and 2 of Theorem 5.2.1 to be *well-formed*. Then, a PIP leading out of some $f$-house $h_0$ with respect to a well-formed matching $M$ is an alternating path $\Pi = \langle h_0, a_0, h_1, a_1, ..., h_x, a_x \rangle$ such that $h_i = f(a_i)$ and $(a_i, h_i) \in M$ for $0 \leq i \leq x$, and $a_i$ prefers $h_{i+1}$ to $h_i$ for $i < x$. A PIP leading out of $h_0$ always exists, which can be seen as follows. Since $h_0$ is an $f$-house and $c_0 \geq 1$, there exists some agent $a'_0 \in f(h_0) \cap M(h_0)$ by Theorem 5.2.1. Then, by definition, $\langle h_0, a'_0 \rangle$ is a PIP leading out of $h_0$. The next lemma shows that any PIP leading out of $h_0$ must contain a sequence of agents with strictly decreasing priorities. Hence, the sequence of agents in $\Pi$ must be distinct since the priority of agents is strictly decreasing.

**Lemma 5.3.1.** *Let $M$ be a well-formed matching. Let $\Pi = \langle h_0, a_0, ..., h_x, a_x \rangle$ be a PIP with respect to $M$ leading out of $h_0$ as defined above. Then, $P(a_{i+1}) < P(a_i)$ for $0 \leq i < x$.*

*Proof.* Let $a_0$ have priority $z_1$. If $x = 0$, then $a_0$ is the last (only) agent in the path. Otherwise, $x > 0$ and it follows by definition of $\Pi$ that $h_0$ is not the first house on $a_0$'s preference list as $h_1$ is a house that $a_0$ prefers to $h_0$. Hence, it must be that $h_1$ is an $f$-house such that $\sum_{p=1}^{z_1-1} f_{p,1} \geq c_1$ by definition of $f(a_0) = h_0$.

Since $M$ is well-formed and $f_1 \geq c_1$, it follows by Theorem 5.2.1 that $|M(h_1)| = c_1$ and $M(h_1) \subseteq f(h_1)$. Now, if $\sum_{p=1}^{z_1-1} f_{p,1} = c_1$, then by definition of an $f$-house, $f_{p,1} = 0$

for $z_1 \leq p \leq k$. Hence, $d_1 \leq z_1 - 1$. Since $f_1 = c_1$, it follows that $M(h_1) \subseteq \bigcup_{p=1}^{z_1-1} f_p(h_1)$ by Theorem 5.2.1. On the other hand, if $\sum_{p=1}^{z_1-1} f_{p,1} > c_1$, then $f_1 > c_1$ and $g_1 \leq z_1 - 1$. It follows by Theorem 5.2.1 again that $M(h_1) \subseteq \bigcup_{p=1}^{z_1-1} f_p(h_1)$. Clearly as a result, $M(h_1) \subseteq \bigcup_{p=1}^{z_1-1} f_p(h_1)$ in all cases.

Since $a_1 \in M(h_1)$, it follows that $f(a_1) = h_1$ and $a_1$ has priority strictly less than $z_1$. Moreover, we can repeat the argument to deduce the priority of each agent $a_i$ in $\Pi$. It is then straightforward to see that the priority of any agent in $\Pi$ must be strictly less than its predecessor so that $P(a_{i+1}) < P(a_i)$ for each $i \geq 0$. □

Let us define the cost of $\Pi$ to be $cost(\Pi) = w(a_x) - w(a_{x-1}) - ... - w(a_0)$ if $x > 0$. Note that $cost(\Pi) = w(a_0)$ if $x = 0$. We now motivate the notion of a PIP as follows. Let us suppose that there exists some agent $a_r$ who prefers $h_0$ to $M(a_r)$. The next lemma shows that any such agent cannot belong to $\Pi$. Now, if $cost(\Pi) < w(a_r)$, we can conclude that the well-formed matching $M$ is not popular because we can promote $a_r$ to $h_j$, and use the PIP to promote each $a_i$ to $h_{i+1}$ for all $i < x$ and demote $a_x$ to $l(a_x)$ to obtain a new matching that is more popular than $M$.

**Lemma 5.3.2.** *Let $M$ be a well-formed matching. Let $\Pi = \langle h_0, a_0, ..., h_x, a_x \rangle$ be a PIP with respect to $M$ leading out of $h_0$ as defined above. Then, any agent $a$ who prefers $h_0$ to $M(a)$ does not belong to $\Pi$.*

*Proof.* Let $a$ have priority $z$. Since $M$ is well-formed, either (i) $M(a) = f(a)$ or (ii) $M(a) = s(a)$. It follows in case (i) that $\sum_{p=1}^{z-1} f_{p,0} \geq c_0$ by definition of $f(a)$. In case (ii), either (a) $h_0 = f(a)$ or (b) $h_0$ is an $f$-house such that $h_0 \neq f(a)$ and $\sum_{p=1}^{z} f_{p,0} \geq c_0$ by definition of $s(a)$. Now, in subcase (a), if $\sum_{p=1}^{z} f_{p,0} < c_0$, then $z \leq d_0$ so that $\bigcup_{p=1}^{z} f_p(h_0) \subseteq \bigcup_{p=1}^{d_0} f_p(h_0) \subseteq M(h_0)$ since $M$ is a well-formed matching. However, this implies that $a \in M(h_0)$, a contradiction. It follows in all cases that $\sum_{p=1}^{z} f_{p,0} \geq c_0$. Using a similar argument as in Lemma 5.3.1, we can establish that $|M(h_0)| = c_0$ and $M(h_0) \subseteq \bigcup_{p=1}^{z} f_p(h_0)$. It follows that $P(a) \geq P(a_0)$ and hence, the priority of $a$ must be greater than the priority of any other agent in $\Pi$ by Lemma 5.3.1. Since $a \neq a_0$, $a$ cannot be an agent in $\Pi$. □

### 5.3.2 Pruning the graph

Let us now introduce Algorithm Prune-WCHA which will enable us to remove certain edges in $G'$ that cannot be part of any popular matching. The algorithm is divided into two

**Algorithm 9** First stage of Algorithm Prune-WCHA

1: **for each** $f$-house $h$ **do**
2: $\lambda(h) := w_1$; {// a suitable upper bound}
3: **for** $z$ in $1..k$ **do**
4: **for each** $a \in P_z$ **do**
5: Let $S$ contain the set of houses that $a$ prefers to $f(a)$;
6: **if** $S \neq \emptyset$ **then**
7: $\lambda_{min}(a, f(a)) := \min\{\lambda(h) : h \in S\}$;
8: **else**
9: $\lambda_{min}(a, f(a)) := \infty$; {// a suitable default value}
10: **if** $\lambda_{min}(a, f(a)) < w_z$ **then**
11: **return** "No popular matching exists";
12: **for each** $f_z$-house $h_j$ **do**
13: $f'_z(h_j) := f_z(h_j)$;
14: **if** $z \leq d_j$ **then**
15: **for each** $a \in f'_z(h_j)$ **do**
16: Remove $(a, s(a))$ from $G'$;
17: **else** {// $z = g_j > d_j$}
18: **for each** $a \in f'_z(h_j)$ such that $\lambda_{min}(a, h_j) < 2w_z$ **do**
19: Remove $(a, h_j)$ from $G'$;
20: Remove $a$ from $f'_z(h_j)$;
21: **if** $f'_z(h_j) = \emptyset$ **then** { // $|f'_z(h_j)| < c_j - \sum_{p=1}^{d_j} f_{p,j}$}
22: **return** "No popular matching exists.";
23: $\lambda_z(h_j) := \min(w_z, \min\{\lambda_{min}(a, h_j) - w_z : a \in f'_z(h_j)\})$;{ // $\lambda_{min}(a, h_j) \geq w_z$}
24: $\lambda(h_j) := \min(\lambda(h_j), \lambda_z(h_j))$;
25: **if** $z > d_j$ **and** $\lambda(h_j) < w_z$ **then**
26: **return** "No popular matching exists.";

stages, with the first stage shown in Algorithm 9 and the second stage shown in Algorithm 10. The first stage is carried out in phases, with each phase corresponding to a priority level $P_z$.

Intuitively, in each phase in the first stage, we compute the costs of PIPs and determine the minimum of these for each $f$-house $h_j$, and then use these values to identify and remove certain edges incident to $f$-houses in $G'$ that cannot belong to any popular matching. Based on the minimum values of PIPs calculated for $f$-houses in the first stage, we then identify and remove in the second stage edges incident to $s$-houses in $G'$ that cannot belong to any popular matching. Let $G''$ denote the graph obtained from $G'$ once the algorithm

**Algorithm 10** Second stage of Algorithm Prune-WCHA

```
1: for each a ∈ A do
2:    Let h_l := s(a);
3:    Let R contain the set of houses that a prefers to h_l;
4:    Let S contain the set of houses that a prefers to f(a);
5:    R := R − (S ∪ {f(a)});
6:    if R ≠ ∅ then
7:       λ_min(a, h_l) := min {λ(h) : h ∈ R};
8:    else
9:       λ_min(a, h_l) := ∞; {// a suitable default value}
10:   if λ_min(a, h_l) < w(a) or f_l ≥ c_l then
11:      Remove (a, h_l) from G';
```

terminates (following these edge removals). The removal of these edges will ensure that any well-formed matching in $G''$ is popular. Over the phases of execution, certain conditions may arise which signal to the algorithm that no popular matching exists.

Recall that $h_j$ may be an $f$-house for more than one priority level, and $h_j$ may be an $f$-house for more than one agent for each priority level. In the algorithm, we will use $\lambda_z(h_j)$ as a variable and its value at the end of the algorithm equals the minimum cost of a PIP leading out of $h_j$ taken over all well-formed matchings in $G''$ such that $(a_r, h_j)$ is the first edge for some $a_r \in P_z$. We will also use $\lambda(h_j)$ to compute the minimum cost taken over all $\lambda_z(h_j)$. Note that we initialise $\lambda(h)$ to $w_1$ for every $f$-house $h$ at the outset of the first stage of Algorithm Prune-WCHA, for if $\Pi$ is any PIP leading out of $h$, then $cost(\Pi) \leq w(a_x)$, where $a_x$ is the final agent on the path. However, $w(a_x) \leq w_1$. Hence, $w_1$ is an upper bound for the final computed value of $\lambda(h)$. Let $\Pi_{min}(h_j)$ denote a PIP with minimum cost leading out of $h_j$ taken over all well-formed matchings in $G''$. Let $cost(\Pi_{min}(h_j))$ denote the cost of this path. Then, as we shall show, the final value of $\lambda(h_j)$ in the execution of the algorithm gives us the value of $cost(\Pi_{min}(h_j))$.

For any agent $a_s \in A$, let $S$ contain the set of houses on $a_s$'s preference list that $a_s$ prefers to $f(a_s)$. Note that $S$ will be empty if $f(a_s)$ is the first house on $a_s$'s preference list. If $S \neq \emptyset$, we will use $\lambda_{min}(a_s, f(a_s))$ within the algorithm to compute the minimum cost of a PIP out of $h_q$, taken over all $h_q \in S$, and over all well-formed matchings in $G''$; otherwise, the algorithm sets $\lambda_{min}(a_s, f(a_s))$ to $\infty$ as a suitable default value. Similarly, let $R$ contain the set of houses on $a_s$'s preference list after $f(a_s)$ that $a_s$ prefers to $s(a_s)$. If $R \neq \emptyset$, we will use $\lambda_{min}(a_s, s(a_s))$ within the algorithm to compute the minimum cost of a

PIP out of $h_q$, taken over all $h_q \in R$, and over all well-formed matchings in $G''$; otherwise, the algorithm sets $\lambda_{min}(a_s, s(a_s))$ to $\infty$ as a suitable default value.

### 5.3.3 Proof of correctness

The following lemma gives us an important technical result regarding the correctness of the algorithm.

**Lemma 5.3.3.** *Suppose that Algorithm Prune-WCHA does not terminate during the execution of its first stage by reporting that no popular matching exists. Let $z$ be an iteration of the for loop on line 3. Let $h_j \in H$ be any $f_z$-house. Then, at the end of this iteration:*

1. *for each $a \in P_z$, if $f(a)$ is not the first ranked house in $a$'s preference list, then $\lambda_{min}(a, f(a))$ equals the minimum cost of all PIPs among all houses that $a$ prefers to $f(a)$ taken over all well-formed matchings in $G''$; else, $\lambda_{min}(a, f(a)) = \infty$.*

2. *$\lambda_z(h_j)$ stores the minimum cost among all PIPs taken over all well-formed matchings in $G''$ such that $(a, h_j)$ is the first edge for some $a \in P_z$.*

3. *$\lambda(h_j)$ stores the minimum cost among all PIPs taken over all well-formed matchings in $G''$ such that $(a, h_j)$ is the first edge for some $a \in P_q$ where $1 \leq q \leq z$.*

4. *if any edge has been removed from $G'$, then it cannot be part of any popular matching.*

*Proof.* Given $1 \leq z \leq k$, we will proceed by induction on $z$.

For the base case, let $z = 1$. If $a \in P_1$, then clearly $S = \emptyset$ for $a$ so that $\infty$ is assigned to $\lambda_{min}(a, f(a))$ as required in line 9. Now, any PIP leading out of $h_j$ and containing the edge $(a, h_j)$ ends at $a$ and has cost $w_1$. Clearly, $w_1$ is assigned to $\lambda_z(h_j)$ as required at line 23 since $\lambda_{min}(a', h_j) = \infty$ for each $a' \in f'_1(h_j)$. Also, $w_1$ is assigned to $\lambda(h_j)$ at line 24 as required, since this is the minimum of $\lambda_z(h_j)$ and the initialised value of $\lambda(h_j)$ which is also $w_1$. Finally, the only edges removed during this iteration are dealt with at lines 15-16 (as the condition in line 18 is not satisfied). For, clearly if $a \in P_1$ and $d_j \geq 1$, $a$ must be assigned to $f(a) = h_j$ and not $s(a)$ in any well-formed matching $M$ by Condition 1 of Theorem 5.2.1. Hence, the edge $(a, s(a))$ cannot belong to any popular matching.

For the inductive case, let us assume that $2 \leq z \leq k$, and that the result is true for $z - 1$. Let $a \in P_z$ be any agent. Suppose that $S \neq \emptyset$. Choose any $h_l \in S$. It follows that $\sum_{p=1}^{z-1} f_{p,l} \geq c_l$ by definition of $h_j = f(a)$. Hence, it is impossible that $h_l$ can be

an $f_p$-house for any $p \geq z$. By the inductive hypothesis, $\lambda(h_l)$ stores the minimum cost among all PIPs leading out of $h_l$ where $(a', h_l)$ is the first edge for some $a' \in P_q$ where $1 \leq q \leq z-1$. Hence, $\lambda(h_l)$ stores the minimum cost among all PIPs leading out of $h_l$ at the end of the iteration $z-1$. Thus, if $S \neq \emptyset$, then when $\lambda_{min}(a, f(a))$ is defined during iteration $z$ in line 7, it contains the minimum cost of a PIP leading out of any house that $a$ prefers to $f(a)$; otherwise, $S = \emptyset$ and $\lambda_{min}(a, f(a))$ is assigned to be $\infty$ in line 9 as required.

Now, it follows that the minimum cost of a PIP out of $h_j$ for which the first edge is $(a, h_j)$ such that $a \in f_z(h_j)$ either stops at $a$ and has cost $w_z$, or it continues. If it continues, it must do so with some edge $(a, h_l)$ such that $a$ prefers $h_l$ to $h_j$. Hence, the minimum cost of a PIP out of $h_j$ for which the first edge is $(a, h_j)$ is the minimum of $w_z$ and $\lambda_{min}(a, h_j) - w_z$. Clearly then, this is exactly the value assigned to $\lambda_z(h_j)$ on line 23 as required. Also, it follows by the inductive hypothesis that $\lambda(h_j)$ should be set at iteration $z$ to be the minimum of $\lambda_z(h_j)$ and the value of $\lambda(h_j)$ at the end of iteration $z-1$. This is precisely the value assigned to $\lambda(h_j)$ at line 24.

Finally, it remains to show that any edge removed during iteration $z$ cannot belong to part of any popular matching. Now, if $z \leq d_j$, then it follows by Theorem 5.2.1 that $a$ must be assigned to $h_j$ and not $s(a)$ for any well-formed matching $M$. Hence, the edge $(a, s(a))$ cannot belong to any well-formed matching and is deleted in line 16 as required. Clearly, if $f_j \leq c_j$, then it is bound to be the case that $z \leq d_j$.

On the other hand, if $z > d_j$, then it follows that in any well-formed matching $M$, $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ but only a proper subset of $f_{g_j}(h_j)$ will be assigned to $h_j$ in $M$. Now, suppose that $a \in M(h_j) \cap f_{g_j}(h_j)$. It follows that $z = g_j$. Let $h_l$ be any house that $a$ prefers to $h_j$, supposing that such a house exists. Clearly, if there exists a minimum cost PIP $\Pi$ out of $h_l$ such that $cost(\Pi_{min}(h_l)) - w_z < w_z$, then $\Pi$ can be used to promote $a$ to $h_l$, and in the process, free up a space in $h_j$ which can thus be assigned to any agent $a'$ in $f_{g_j}(h_j) \backslash M(h_j)$. Clearly, $M(a') = s(a')$ since $M$ is well-formed so that $a'$ improves as result. It follows that $M$ cannot be popular since we can promote $a'$ to $h_j$, promote $a$ to $h_l$ and promote along $\Pi$ to obtain a more popular matching than $M$. Hence, if $\lambda_{min}(a, h_j) < 2w_z$, then $M$ is not popular. Since $M$ is arbitrary, the edge $(a, h_j)$ cannot belong to any popular matching so that we delete it in line 19.

Note that $\Pi_{min}(h_l)$ must be a minimum cost PIP with respect to $M$. For, let us consider the first edge $(b, h_l)$ in $\Pi_{min}(h_l)$. Note that $f_l \geq c_l$ and $g_l < z$ since $h_l$ is a house

that $a$ prefers to $f(a) = h_j$.

Suppose firstly that $b \in f_{g_l}(h_l)$. Let $\lambda_{d_l}$ be the value of $\lambda(h_l)$ at the end of phase $g_l - 1$. Now, we have that the value of $\lambda(h_l)$ as computed in phase $g_l$ by lines 23-24 of the algorithm is equal to $\min(w_{g_l}, \lambda_{d_l}, \min\{\lambda_{min}(b', h_l) - w_{g_l} : b' \in f'_{g_l}(h_l)\})$. Let us suppose that $\min\{\lambda_{min}(b', h_l) - w_{g_l} : b' \in f'_{g_l}(h_l)\} < w_{g_l}$. Then, there exists some agent $b' \in f'_{g_l}(h_l)$ such that $\lambda_{min}(b', h_l) - w_{g_l} < w_{g_l}$, i.e. $\lambda_{min}(b', h_l) < 2w_{g_l}$. However, such a $b'$ would have been removed from $f'_{g_l}(h_l)$ at line 20, a contradiction. Hence, $\lambda_{min}(b', h_l) - w_{g_l} \geq w_{g_l}$ for all $b' \in f'_{g_l}(h_l)$. It follows that any minimum cost PIP in $G''$ (with respect to any well-formed matching) with $(b', h_l)$ as its first edge must have cost greater than or equal to $w_{g_l}$, i.e. $cost(\Pi_{min}(h_l)) \geq w_{g_l}$. Now, suppose that $\lambda_{d_l} < w_{g_l}$. Then, there exists a PIP leading out of $h_l$ whose first edge is $(c, h_l)$ where $P(c) \leq d_l$, with cost less than $w_{g_l}$. However, this then contradicts the fact that the PIP with $(b, h_l)$ as its first edge has minimum cost for $h_l$ as we supposed. Hence, $w_{g_l}$ is a lower bound for the final computed value of $\lambda(h_l)$. Clearly then, $\lambda(h_l) = w_{g_l}$. Since $(b, h_l)$ is the first edge of $\Pi_{min}(h_l)$ where $b \in f_{g_l}(h_l)$, then as this path is defined with respect to some well-formed matching, it follows that $(b', h_l) \in M$ for some $b' \in f_{g_l}(h_l)$ (possibly $b = b'$), since $M$ is well-formed. Then, $\langle h_l, b' \rangle$ is a PIP of cost $w_{g_l}$ with respect to $M$. Moreover, since $w_{g_l} = cost(\Pi_{min}(h_l)) < 2w_z$ as established in the previous paragraph, it follows that we can promote $a$ to $h_l$, promote $a'$ to $h_j$ and demote $b'$ from $h_l$ so that $M$ is not popular as shown above.

Hence, $b \in \bigcup_{p=1}^{d_l} f_p(h_l)$. Clearly then, $(b, h_l)$ must belong to every well-formed matching by Condition 1(a) of Theorem 5.2.1 so that $(b, h_l)$ must belong to $M$. It follows that we can repeat the above argument to show that $\Pi_{min}(h_l)$ is a minimum cost PIP with respect to $M$ by considering the remaining alternate edges in $\Pi_{min}(h_l)$. If each alternate edge $(c, h_x)$ satisfies the condition $c \in \bigcup_{p=1}^{d_x} f_p(h_x)$, then the result is immediate. Otherwise, it must be the case that we encounter some edge $(c', h_{x'})$ in $\Pi_{min}(h_l)$ such that $c' \in f_{g_{x'}}(h_{x'})$. Clearly then, $(c', h_{x'})$ is the final edge in $\Pi_{min}(h_l)$ so that we must be able to promote $a$ to $h_l$, promote $a'$ to $h_j$ and promote along $\Pi_{min}(h_l)$ to obtain a more popular matching than $M$ by a similar argument to that in the previous paragraph. □

The next three lemmas establish the correctness of the algorithm.

**Lemma 5.3.4.** *Suppose that Algorithm Prune-WCHA does not terminate during the execution of its first stage by reporting that no popular matching exists. Then, any edge removed by Algorithm Prune-WCHA over both stages cannot belong to a popular matching.*

*Proof.* By Lemma 5.3.3, any edges removed by Algorithm Prune-WCHA in the first stage cannot belong to any popular matching. We now show that any edges removed by the algorithm in the second stage also cannot belong to any popular matching.

Let $M$ be any well-formed matching. Let $a$ be any agent and let $P(a) = z$. Also, let $R$ contain the set of houses between $f(a)$ and $s(a)$ on $a$'s preference list that $a$ prefers to $s(a)$ (not including $f(a)$ and $s(a)$). Let $s(a) = h_l$. Suppose that $M(a) = h_l$. Let $h_j \in R$ and suppose that $cost(\Pi_{min}(h_j)) < w_z$. Clearly, $\Pi_{min}(h_j)$ must be a minimum cost PIP with respect to $M$ by a similar argument to that used in the proof of Lemma 5.3.3. Then, $\Pi_{min}(h_j)$ can be used to free up $h_j$ and promote $a$ to $h_j$ to obtain a more popular matching than $M$. Hence, $M$ cannot be popular. It follows that an edge pruned due to the first condition in line 10 of the second stage of the algorithm cannot belong to any popular matching.

Now, if $f_l \geq c_l$ and $M(a) = h_l$, then $M$ cannot be popular by Condition 1 of Theorem 5.2.1, since $M(h_l) \not\subseteq \bigcup_{p=1}^{g_l} f_p(h_l)$. This shows that the edge $(a, h_l)$ pruned due to the second condition in line 10 of the second stage of the algorithm also cannot belong to any popular matching.

It thus follows that any edges removed by the algorithm cannot belong to a popular matching. □

**Lemma 5.3.5.** *If Algorithm Prune-WCHA reports that no popular matching exists, then there does not exist any well-formed matching in $G'$ that is popular.*

*Proof.* Let us consider the cases where Algorithm Prune-WCHA reports that no popular matchings exist as a result of some condition being satisfied: (i) lines 10-11 ,(ii) lines 21-22 and (iii) lines 25-26 of the first stage respectively. Let $a$ be any agent and let $P(a) = z$. Also, let $f(a) = h_j$.

In case (i), let us suppose that $M(a) = h_j$ for some well-formed matching $M$. Let $h_l$ be a house that $a$ prefers to $h_j$ such that $\lambda_{min}(a, h_j) = cost(\Pi_{min}(h_l))$. It follows by a similar argument to that used in the proof of Lemma 5.3.3 that $\Pi_{min}(h_l)$ must be a minimum cost PIP with respect to $M$. Now, if $\lambda(h_l) < w_z$, then we can use $\Pi_{min}(h_l)$ to free $h_l$ and

then promote $a$ to $h_l$ to obtain a more popular matching than $M$. Hence, $M$ cannot be popular. Since $M$ is arbitrary, it follows that no popular matching exists.

In case (ii), clearly $f_j > c_j$. Now, if $f'_{g_j}(h_j) = \emptyset$ after the removal of edges in lines 18-20, then it follows that no well-formed matching can exist in $G''$ since no matching can satisfy Condition 1(b) of Theorem 5.2.1. Hence, no popular matching can exist.

In case (iii), let us suppose that $z = g_j$. Clearly, only a proper subset of agents in $f_{g_j}(h_j)$ can be assigned to $h_j$ in $M$ since $f_j > c_j$. Let $a \in f_{g_j}(h_j) \backslash M(h_j)$. Note that $\Pi_{min}(h_j)$ must be a minimum cost PIP with respect to $M$ using a similar argument in the proof of Lemma 5.3.3. Now, if $\lambda(h_j) < w_{g_j}$, $\Pi_{min}(h_j)$ can be used to free up a place in $h_j$ and then promote $a$ (who must be assigned to $s(a)$ in $M$) to $h_j$ to obtain a matching that is more popular than $M$. Since $M$ is arbitrary, no popular matching exists. □

**Lemma 5.3.6.** *Suppose that Algorithm Prune-WCHA does not state that no popular matching exists. Let $M$ be a well-formed matching in the pruned graph $G''$. Then, $M$ is popular.*

*Proof.* Now, if $M$ is not popular, it follows that there exists another matching $M'$ which is more popular than $M$. Let us clone $G''$ to obtain a cloned graph $C(G'')$ as follows. We replace every house $h_j \in H$ with the clones $h_j^1, h_j^2, \ldots, h_j^{c_j}$. We then divide the capacity of each house among its clones by allowing each clone to have capacity 1. In addition, if $(a, h_j)$ is an edge in $G''$, then we add $(a, h_j^p)$ to the edge set of $C(G'')$ for all $p$ $(1 \leq p \leq c_j)$. Let us then adapt the well-formed matching $M$ in $G''$ to obtain its clone $C(M)$ in $C(G'')$ as follows. If a house $h_j$ in $G''$ is assigned to $x_j$ agents $a_1, ...a_{x_j}$ in $M$, then we add $(a_p, h_j^p)$ to $C(M)$ for $1 \leq p \leq x_j$, so that $|C(M)| = |M|$. We repeat a similar process for $M'$ to obtain its clone $C(M')$ in $C(G'')$.

Let us consider $X = C(M) \oplus C(M')$. Since $sat(M', M) > 0$, let $a \in A$ be an agent who prefers $M'$ to $M$. Let $P(a) = z$ and let $M'(a) = h_j$. We will show that there exists a PIP $\Pi$ leading out of $h_j$ with respect to $M$. Since $M$ is well-formed, we can reuse a similar argument to the proof of Lemma 5.3.2 to establish that $h_j$ is an $f$-house such that $\sum_{p=1}^{z} f_{p,j} \geq c_j$. It follows that $h_j$ is full in $M$ and $M(h_j) \subseteq f(h_j)$ by Theorem 5.2.1. Let $a_r \in M(h_j) \backslash M'(h_j)$ ($a_r$ must exist since $h_j$ is full in $M$) and let $P(a_r) = z_1$. Then, $a \neq a_r$. Also, it follows that $f(a_r) = h_j$ and $z_1 \leq z$. If $a_r$ does not prefer $M'$ to $M$, then we finish tracing $\Pi$. Otherwise, we will extend $\Pi$ to make sure that it ends with some agent $b$ who prefers $M$ to $M'$. It follows by definition of $f(a_r)$ that $M'(a_r) = h_l$ is

an $f$-house that $a_r$ prefers to $h_j$ such that $\sum_{p=1}^{z_1-1} f_{p,l} \geq c_l$ and hence by Theorem 5.2.1, $M(h_l) \subseteq f(h_l)$. Let $a_s \in M(h_l) \backslash M'(h_l)$ and let $P(a_s) = z_2$. Clearly then, $z_2 < z_1$. It follows by the same argument as for $a_r$ that if $a_s$ does not prefer $M'$ to $M$, then we finish tracing $\Pi$, i.e. $\Pi = \langle h_j, a_r, h_l, a_s \rangle$. Otherwise, we repeat the argument until we encounter an agent $a_t$ who does not prefer $M'$ to $M$ so that $\Pi$ terminates. Clearly, this will eventually happen since all agents in $\Pi$ are assigned in $M$ to their $f$-house and the priority levels of agents are strictly decreasing so that we must eventually reach some agent $a_t \in P_1$ such that $M(a_t) = f(a_t)$. However, it is then impossible that $a_t$ prefers $M'$ to $M$. Finally, by construction of $\Pi$, it follows that $\Pi$ belongs to $X$ since $\Pi$ (with appropriate superscripts for house clones) consists of alternate edges in $C(M) \backslash C(M')$ and $C(M') \backslash C(M)$.

We have established that for every $a \in P(M', M)$, there exists a PIP $\Pi(a)$ leading out of $h_j$, where $h_j = M'(a)$. Let $\Gamma = \{\Pi(a) : a \in P(M', M)\}$ and let $\Gamma' \subseteq \Gamma$ contain only those maximal PIPs in $\Gamma$. We will show that there exists an agent $d \in A$ such that $\Pi(d) \in \Gamma'$ and $cost(\Pi(d)) < w(d)$. For, suppose that $cost(\Pi(a)) \geq w(a)$ for every $\Pi(a) \in \Gamma'$. Let $\Pi(a) \in \Gamma'$ and let $\Pi(a) = \langle h_0, a_0, h_1, a_1, ..., h_x, a_x \rangle$. We define $l(\Pi(a)) = a_x$. Also, $cost(\Pi(a)) = w(a_x) - w(a_{x-1}) - ... - w(a_0) \geq w(a)$, i.e. $w(a) + w(a_0) + ... + w(a_{x-1}) \leq w(a_x)$. Now, $\{a, a_0, ..., a_{x-1}\} \subseteq P(M', M)$ whilst $a_x \in P(M, M')$. Let $D$ be the connected component of $X$ containing $\Pi(a)$ (with appropriate superscripts for house clones). It follows that $D$ must be a path or cycle whose edges alternate between $C(M)$ and $C(M')$. Clearly, $D$ cannot be an even-length alternating path with more agents than houses, or an odd-length alternating path whose end edges belong to $C(M')$, for otherwise we have an agent who is unassigned in $C(M)$ and hence in $M$, a contradiction to the definition of a well-formed matching. Hence, $D$ is either an (i) even-length alternating path with more houses than agents, or (ii) an odd-length alternating path whose end edges belong to $C(M)$, or (iii) a cycle. It is obvious that $D$ contains distinct agents and so we cannot have overlapping maximal PIPs. Hence, by construction of $\Gamma'$, the agents in $\Pi(a)$, together with $a$, but not including $l(\Pi(a))$, taken over all $\Pi(a) \in \Gamma'$, form a partition of $P(M', M)$. Moreover, for every such $a$, we have established the existence of some $l(\Pi(a)) \in P(M, M')$. Hence,

$$\begin{aligned}\sum_{a\in P(M',M)} w(a) &= \sum_{\Pi(a)\in\Gamma'} w(a) + \sum_{\Pi(a)\in\Gamma'} \sum \big\{w(a') : a' \in \Pi(a) \wedge a' \neq l(\Pi(a)\big\} \\ &\leq \sum_{\Pi(a)\in\Gamma'} \big\{w(a') : a' = l(\Pi(a))\big\} \\ &\leq \sum_{a\in P(M,M')} w(a)\end{aligned}$$

It follows that $sat(M', M) \leq 0$, a contradiction. As a result, $cost(\Pi(d)) < w(d)$ for some $\Pi(d) \in \Gamma'$. Let $h_j = M'(d)$. Now, if $M(d) = f(d)$, then lines 10-11 of the first stage of the algorithm would report that no popular matching exists since $\lambda_{min}(d, f(d)) < w(d)$, a contradiction. Hence, $M(d) = s(d)$ and $h_j$ is (i) better than $f(d)$, or (ii) equal to $f(d)$, or (iii) between $f(d)$ and $s(d)$ on $a$'s preference list. In case (i), we obtain the same contradiction as when $M(d) = f(d)$ since $\lambda_{min}(d, f(d)) < w(d)$. In case (ii), $f(d) = h_j$. Since $M(d) = s(d)$, it must be the case that $d \in f_{g_j}(h_j)$ for otherwise $(d, s(d))$ would have been deleted by line 16 of the algorithm. Clearly though, lines 25-26 of the first stage of the algorithm would report that no popular matching exists, a contradiction. In case (iii), $(d, s(d))$ would have been deleted by lines 10-11 of the second stage of the algorithm since $\lambda_{min}(d, s(d)) < w(d)$, a contradiction. It follows that we obtain a contradiction in all cases so that $M'$ is not more popular than $M$. □

Finally, the next lemma shows that if there is no well-formed matching in the graph $G''$, then no popular matching exists.

**Lemma 5.3.7.** *Let $G''$ be the reduced graph of a given* WCHA *instance $I$. If there is no well-formed matching in $G''$, then no popular matching exists in $I$.*

*Proof.* Suppose that there exists a popular matching $M$ in $I$. Now, by Theorem 5.2.1, $M$ is a well-formed matching in $G'$. Moreover, all edges of $M$ must belong to $G''$ by Lemma 5.3.4. However, this implies that $M$ is a well-formed matching in $G''$, a contradiction. □

We now use the example in Figure 5.1 to illustrate our algorithm. After the first stage, we have $\lambda(h_1) = 7$, $\lambda(h_3) = 3$ and $\lambda(h_4) = 2$. We remove the edges $(a_1, h_2)$ in phase 1, and $(a_2, h_4)$ and $(a_3, h_5)$ in phase 2 of the first stage (in line 16 of the first stage) since $a_1$ belongs to $f_{d_1}(h_1)$, and $a_2$ and $a_3$ belong to $f_{d_3}(h_3)$ respectively. We also remove the edge $(a_4, h_4)$ in phase 3 of the first stage (in lines 19-20 of the first stage) since $\lambda_{min}(a_4, h_4) = 3 < 2w(a_4)$. No further edges are removed in the second stage.

### 5.3.4 Finding a popular matching

We are now left with the task of finding a well-formed matching $M$ in $G''$ in order to find a popular matching if one exists. Note that the removal of edges from $G'$ by Algorithm Prune-WCHA effectively reduces the problem to that of finding a popular matching in an instance of CHA. For let us consider the problem of trying to assign agents to each $f$-house $h_j$ so that $h_j$ satisfies Condition 1 of a well-formed matching.

Now, if $f_j \leq c_j$, then ensuring that $\bigcup_{p=1}^{d_j} f_p(h_j) \subseteq M(h_j)$ is equivalent to ensuring Condition 1(a) of Theorem 4.2.1 on page 47. This work is done by lines 2-8 of Algorithm Popular-CHA. On the other hand, if $f_j > c_j$, we need to ensure that those agents with the correct priorities are assigned to $h_j$ in $M$, i.e. there does not exist any agent $a \in \bigcup_{p=1}^{d_j} f_p(h_j) \backslash M(h_j)$. Now, since line 16 in the first stage of Algorithm Prune-WCHA ensures the removal of the edge $(a, s(a))$ of every such $a$ where $a \in \bigcup_{p=1}^{d_j} f_p(h_j)$, $a$ must be assigned to $f(a)$ if an *agent-complete* matching (i.e. a matching in which all agents are assigned) is to exist. This is equivalent to the work done by lines 10-12 of Algorithm Popular-CHA on page 48, which tries to find an agent-complete matching and reports that no popular matching exists if unsuccessful. Furthermore, lines 15-18 of Algorithm Popular-CHA also ensures that if $f_j > c_j$, then $|M(h_j)| = c_j$ and $M(h_j) \backslash \bigcup_{p=1}^{d_j} f_p(h_j) \subseteq f_{g_j}(h_j)$. Lastly, we need to ensure that each agent is assigned to either $f(a)$ or $s(a)$ and it is evident that running Algorithm Popular-CHA on $G''$ does this. Hence, we can find a popular matching in WCHA, if one exists, by running Algorithm Popular-CHA on the reduced graph $G''$. As illustration, if we run Algorithm Popular-CHA on the example in Figure 5.1 after edge removals through Algorithm Prune-WCHA, then Algorithm Popular-CHA will return the following matching $M = \{(a_1, h_1), (a_2, h_3), (a_3, h_3), (a_4, h_5), (a_5, h_4), (a_6, h_4)\}$ which may be verified to be popular.

Let us now consider the time taken to find a popular matching in an instance of WCHA, or to report that no such matching exists. First of all, it takes $O(m)$ time to define the $f$- and $s$-houses. Let us then consider the time complexity of Algorithm Prune-WCHA. It is clear that the subgraph $G'$ can be constructed in $O(m)$ time and has $O(n_1)$ edges since each agent has degree 2 in $G'$. Clearly, in the first stage of the algorithm, initialising $\lambda(h_j)$ for each $f$-house takes $O(n_2)$ time. Next, we iterate over every agent $a$ to define $\lambda_{min}(a, f(a))$. In order to do so, we traverse the preference list of $a$ to find the minimum cost of all PIPs among all houses that $a$ prefers to $f(a)$, if such houses exist. Even though this occurs in phases, with the total number of phases equal to the number of

priority levels, the computation time for this takes $O(m)$ time overall by the total length of preference lists. Hence, defining $\lambda_{min}(a, f(a))$ for every agent $a$ takes $O(m)$ time overall.

In order to define $\lambda_z(h_j)$ (and hence $\lambda(h_j)$) for each $f$-house $h_j$, we need to iterate over every agent $a$ such that $a \in f_z(h_j)$. Again, the time complexity for this is bounded by the total length of preference lists so that it takes $O(m)$ time overall to define $\lambda_z(h_j)$ (and hence $\lambda(h_j)$) for each $f$-house and to remove those edges which cannot belong to any popular matching (in lines 16 and 19-20 of the first stage of the algorithm). By a similar argument, the second stage of the algorithm also takes $O(m)$ time so that Algorithm Prune-WCHA takes $O(m)$ time overall. Now, it takes $O(\sqrt{C}n_1+m)$ time, using Algorithm Popular-CHA, to find a well-formed matching (if one exists) in $G''$, where $C$ is the total capacity of the houses. It follows that we obtain the following results for the time complexity of finding a popular matching in WCHA.

**Theorem 5.3.1.** *Let $I$ be an instance of* WCHA. *Then, we can find a popular matching in $I$, or determine that none exists, in $O(\sqrt{C}n_1 + m)$ time.*

### 5.3.5 Finding a maximum popular matching

It remains to consider the problem of finding a maximum popular matching in WCHA. Let us run Algorithm Label-f and Algorithm Prune-WCHA as before to define $f$- and $s$-houses and to delete certain edges which cannot belong to any popular matching. We then adopt a similar algorithm to that in Section 4.2.3 on page 49 for the analogous problem in CHA as follows.

That is, let $A_1$ be the set of all agents $a$ with $s(a) = l(a)$, and let $A_2 = A\backslash A_1$. Our objective is to find a well-formed matching in $G''$ which minimises the number of $A_1$-agents who are assigned to their last resort house. We let $A'$ denote the set $\left\{a \in \bigcup_{p=1}^{d_j} f_p(h_j) : h_j \in H\right\}$. We begin by carrying out a pre-processing step on $G''$ to compute a matching $M_0$ that assigns each agent in $A'$ to his $f$-house. We then try to find a maximum matching $M'$ in $G''$ that only involves the $A_2\backslash A'$ agents and their incident edges. If $M'$ is not an agent-complete matching of $A_2\backslash A'$ agents, then clearly $I$ admits no popular matching. Otherwise, we remove all edges in $G''$ that are incident to a last resort house, and try to assign additional $A_1\backslash A'$-agents to their $f$-houses by repeatedly finding an augmenting path with respect to $M'$ using Gabow's algorithm [15] in a similar approach to that for CHA in Section 4.2.3. Let $M''$ be the matching obtained by augmenting $M'$. If any $A_1$-agent remains unassigned at the end of this step, we simply assign

him to his last resort house, to obtain an agent-complete matching of $A\backslash A'$ agents in $G''$. Let $M = M_0 \cup M''$. If any agent $a$ belonging to $A\backslash A'$ is not assigned to his $f$-house $h_j$ but $h_j$ is undersubscribed in $M$, we promote $a$ from $M(a)$ to $h_j$. Then, clearly $M$ will be a well-formed matching in $G''$, and hence popular by Lemma 5.3.6. It follows that $M$ is a maximum popular matching, giving the following theorem.

**Theorem 5.3.2.** *Given an instance of* WCHA*, we can find a maximum popular matching, or determine that none exists, in* $O(\sqrt{C}n_1 + m)$ *time.*

### 5.3.6 "Cloning" versus our direct approach

A straightforward solution to finding a popular matching, given an instance $I$ of WCHA, may be to use "cloning" to create an instance $J$ of WHAT, and then to apply the $O(\min(k\sqrt{n}, n)m)$ algorithm of [43] to $J$. Firstly, we create $c_j$ clones $h_j^1, h_j^2, ..., h_j^{c_j}$ of each house $h_j$ in $I$, where each clone has a capacity of 1. In addition, we replace each occurrence of $h_j$ in a given agent's preference list with the sequence $h_j^1, h_j^2, ..., h_j^{c_j}$, the elements of which are listed in a single tie at the point where $h_j$ appears. Let $G_J$ denote the underlying graph of $J$. Then, $G_J$ contains $n' = n_1 + C$ nodes. For each $a_i \in A$, let $A_i$ denote the set of acceptable houses for $a_i$, and let $c_{min} = \min\{c_j : h_j \in H\}$. Then the number of edges in $G_J$ is $m' = \sum_{a_i \in A} \sum_{h_j \in A_i} c_j \geq mc_{min}$. Hence, the complexity of applying the algorithm of [43] to $J$ is $\Omega(\min(k\sqrt{n_1 + C}, n_1 + C)mc_{min})$. Now, the complexity of our algorithm may be rewritten as $O(\sqrt{C}n_1)$ or $O(m)$ depending on which component dominates the running time. If $n_1 + C \geq k\sqrt{n_1 + C}$, then the cloning approach takes $\Omega(k\sqrt{n_1 + C}mc_{min}))$ time which is slower than our algorithm by a factor of $\Omega(kc_{min})$. Otherwise, if $n_1 + C < k\sqrt{n_1 + C}$, then the cloning approach takes $\Omega(mc_{min}(n_1 + C))$ time which is slower than our algorithm by a factor of $\Omega(\sqrt{n_1 + C}c_{min})$. It follows that the cloning method is slower than our direct approach for all possible cases.

## 5.4 Open problem

We conclude with the following open problem. Suppose that we are presented with an instance $J$ of WCHA in which the preference lists of agents are allowed to contain ties, i.e. an instance of WCHAT. Is the problem of finding a popular matching (or reporting that none exists) in $J$ then solvable in polynomial time?

# Chapter 6

# Popular matchings in SMTI-SYM

## 6.1 Chapter overview

The classical Stable Marriage problem and its variants involving ties and incomplete preference lists were introduced in Section 1.4.1. In this chapter, we study popular matchings in a special case of SMTI in which preference lists are symmetric (SMTI-SYM). An instance $I$ of SMTI is said to have *symmetric preferences* when the rank (to be defined formally later) of each man $u$ on a woman $w$'s preference list is equal to that of $w$ on $u$'s preference list for any (man,woman) pair $(u, w)$. Little is known about how to find maximum popular matchings in matching problems where all participants have preferences (i.e. all participants are agents). This chapter presents the first known characterisation of popular matchings in the bipartite setting with preferences on both sides. We remark that our characterisation could form the basis of a polynomial-time algorithm for finding a maximum popular matching in the context of SMTI-SYM as well as other matching problems in which all the participating agents involved have preferences, e.g. SMTI and SRTI.

The main results of this chapter, and their organisation are as follows. We give some terminology and preliminary results on popular matchings in SMTI-SYM in Section 6.2. We next present necessary conditions for a matching to be popular given a SMTI-SYM instance $I$ in Section 6.3. We then develop an insight into the underlying structure of the problem in Section 6.4 where we introduce what are known as *mutually exclusive edge pairs*. Together with the results of Section 6.3, we obtain necessary and sufficient conditions for a matching to be popular in a given SMTI-SYM instance. Finally, we show how to use this characterisation to provide an efficient means of testing if a given

matching is popular in $O(\sqrt{n}m)$ time where $n$ is the total number of men and women and $m$ is the total length of preference lists in $I$.

## 6.2 Basic terminology and preliminary results

An instance $I$ of SMTI-SYM comprises two disjoint sets $U$ and $W$, where $U = \{u_1, ..., u_{n_1}\}$ is the set of men and $W = \{w_1, ..., w_{n_2}\}$ is the set of women. Each man $u \in U$ ranks (strictly or with ties) a subset of $W$ (the *acceptable* women for $u$) represented by his *preference list* and vice versa. Let $a \in U \cup W$ be any agent. If $a$'s preference list contains the agent $a'$, then we say that $a'$ is an *acceptable partner* for $a$. Let the bipartite graph $G = (U, W, E)$ be the *underlying graph* of $I$, where we let the edge set $E$ of $G$ represent the acceptable partners of the agents.

Given a man $u \in U$ and an acceptable woman $w \in W$ for $u$, we define $rank_u(w)$ to be the number of agents that $u$ prefers to $w$ plus 1, and vice versa. If $rank_u(w) = k$, we say that $w$ is a *kth choice* of $u$. Moreover, if $rank_u(w) = k$, then $rank_w(u) = k$, and we say that the preference lists are *symmetric*.

We create a unique *last resort* partner $l(a)$ for each $a$ and append $l(a)$ to $a$'s preference list. We let every last resort agent $l(a)$ have a preference list that contains only $a$. We also let $z$ be the maximum length taken over all preference lists, including last resorts. Note that to enforce the symmetry of the preference lists with the introduction of last resort partners, we let $rank_{l(a)}(a) = rank_a(l(a))$ for each agent $a$ even though the preference list of $l(a)$ has only size one. We also henceforth assume that $G$ contains the vertex $l(a)$ and the edge $(a, l(a))$ for each $a \in U \cup W$, and that $U$ and $W$ contain the respective last resort men and women. We let $n = n_1 + n_2$ and $m = |E|$.

We assume the definition of a matching in $I$ as defined for a given instance of SM in Section 1.4.1. Given two matchings $M$ and $M'$ in $I$, we say that an agent $a$ *prefers* $M$ to $M'$ if either (i) $a$ is assigned in $M$ and unassigned in $M'$, or (ii) $a$ is assigned in both $M$ and $M'$ and prefers $M(a)$ to $M'(a)$. Let $P(M, M')$ denote the set of non last-resort agents[1] who prefer $M$ to $M'$. Then, the *satisfaction* of $M$ with respect to $M'$ is defined as $sat(M, M') = P(M, M') - P(M', M)$. We say that $M$ is *more popular than* $M'$, denoted by $M \succ M'$, if $sat(M, M') > 0$. Furthermore, a matching $M$ in $I$ is *popular* if there is no other matching in $I$ that is more popular than $M$.

[1]We do not allow last resort agents to contribute to $P(M, M')$ for any two matchings $M$ and $M'$.

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: $w_1$ $w_2$ $w_3$ | $w_1$: ($u_1$ $u_2$ $u_3$) |
| $u_2$: $w_1$ $w_2$ $w_3$ | $w_2$: $u_4$ ($u_1$ $u_2$ $u_3$) |
| $u_3$: $w_1$ $w_2$ $w_3$ | $w_3$: $u_4$ $u_5$ ($u_1$ $u_2$ $u_3$) |
| $u_4$: ($w_2$ $w_3$ $w_4$) | $w_4$: $u_4$ |
| $u_5$: $w_5$ $w_3$ | $w_5$: $u_5$ |

Figure 6.1: An instance $I_1$ of SMTI-SYM

For the remainder of this section, we make several preliminary observations concerning popular matchings in SMTI-SYM.

First of all, an instance of SMTI-SYM need not admit a popular matching. For, consider instance $I_1$ in Figure 6.1. In any popular matching $M$ in $I_1$, it must be the case that $(u_5, w_5) \in$M. For, suppose not. Then, we can promote $u_5$ and $w_5$ to each other (and demote $M(u_5)$ to $l(M(u_5))$ if $M(u_5) \neq l(M(u_5)))$ to obtain a more popular matching than $M$, a contradiction. It must also be the case that $(u_4, w_4) \in M$. For, otherwise suppose that $(u_4, w_3) \in M$. Then, it follows that $(u_i, l(u_i)) \in M$ for some $i (1 \leq i \leq 3)$. However, we can then unassign $u_4$ from $w_3$ and promote $u_i$ to $w_3$, and promote $w_4$ to $u_4$ to obtain a more popular matching than $M$, a contradiction. Note that a similar contradiction is obtained if $(u_4, w_2) \in M$ instead. Hence, it must be the case that $(u_5, w_5)$ and $(u_4, w_4)$ belong to $M$. It follows that $M$ can only be one of the following matchings:

$$\begin{aligned} M_1 &= \{(u_1, w_1), (u_2, w_2), (u_3, w_3), (u_4, w_4), (u_5, w_5)\} \\ M_2 &= \{(u_1, w_2), (u_2, w_3), (u_3, w_1), (u_4, w_4), (u_5, w_5)\} \\ M_3 &= \{(u_1, w_3), (u_2, w_1), (u_3, w_2), (u_4, w_4), (u_5, w_5)\} \end{aligned}$$

However, it is straightforward to verify that none of these matchings is popular since $M_1 \succ M_2 \succ M_3 \succ M_1$, the problem being that the more popular than relation is not acyclic. We next note that popular matchings in SMTI-SYM can have different cardinalities, as seen in instance $I_2$ in Figure 6.2. Here, $M_4 = \{(u_2, w_1), (u_3, w_2)\}$ is a popular matching of cardinality 2. However, the unique maximum popular matching is $M_5 = \{(u_1, w_1), (u_2, w_2), (u_3, w_3)\}$ which has cardinality 3.

We also observe that the cardinality of a maximum popular matching can be smaller than that of a maximum matching. For, consider instance $I_3$ in Figure 6.3. Here, $M_6 = \{(u_1, w_1), (u_2, w_2), (u_3, w_3), (u_4, w_4)\}$ is the unique maximum matching in $I_3$ which has

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: $w_1$ | $w_1$: ($u_1$ $u_2$) |
| $u_2$: $w_1$ $w_2$ | $w_2$: $u_3$ $u_2$ |
| $u_3$: ($w_2$ $w_3$) | $w_3$: $u_3$ |

Figure 6.2: An instance $I_2$ of SMTI-SYM

cardinality 4. However, $M_6$ is not popular because $M_7 = \{(u_1, w_2), (u_2, w_3), (u_3, w_4)\}$ is a matching that is more popular than $M_6$.

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: ($w_1$ $w_2$) | $w_1$: $u_1$ |
| $u_2$: $w_3$ $w_2$ | $w_2$: $u_1$ $u_2$ |
| $u_3$: $w_4$ $w_3$ | $w_3$: $u_2$ $u_3$ |
| $u_4$: $w_4$ | $w_4$: ($u_3$ $u_4$) |

Figure 6.3: An instance $I_3$ of SMTI-SYM

Finally, we remark that in the context of SMI, it may be shown that a stable matching is also popular [31]. Recall from Section 1.4.1.1 that the extended Gale-Shapley algorithm always finds a stable matching in any given SMI instance. Hence, the algorithm also provides a way for finding a popular matching in any given SMI instance $I$. However, this is not true in the case of SMTI. First of all, recall from Section 1.4.1.3 that only weakly stable matchings are guaranteed to exist in any given SMTI instance (as opposed to strongly stable or super-stable matchings). However, weak stability need not imply popularity in the context of SMTI. For, consider instance $I_4$ in Figure 6.4. Then, it may be verified that the matching $M_8 = \{(u_1, w_2), (u_2, w_1)\}$ is a weakly stable matching. However, $M_8$ is not popular because $M_9 = \{(u_1, w_1), (u_2, w_2)\}$ is more popular than $M_8$. Furthermore, weakly stable matchings can have different cardinalities, and a given weakly stable matching could be smaller than the size of a maximum popular matching. Indeed, as mentioned in Section 1.4.1.3, the problem of finding a maximum weakly stable matching given an SMTI instance is NP-hard. These two issues motivate the need to find an alternative algorithm if we want to find a popular matching of maximum cardinality given any SMTI instance.

This chapter provides the first step towards finding an algorithm that will construct a maximum popular matching given an SMTI instance, if one exists, by providing a

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: $w_1$ $w_2$ | $w_1$: ($u_1$ $u_2$) |
| $u_2$: ($w_1$ $w_2$) | $w_2$: $u_2$ $u_1$ |

Figure 6.4: An instance $I_4$ of SMTI-SYM

characterisation of popular matchings that could be used by any approach to solve the problem in the restricted case of SMTI-SYM.

## 6.3 Characterising popular matchings

For each agent $a \in U \cup W$, let $f(a)$ denote the highest ranking set of agents on $a$'s preference list. We call any agent belonging to $f(a)$ an *f-partner* of $a$. Define $E_1 = \{(u, w) : u \in U \wedge w \in f(u)\}$ to be the set of *first-choice edges* of $G$. Define also the *first-choice graph* of $G$ as $G_1 = (U, W, E_1)$. Note that it is trivial to find the unique popular matching given any instance of the Stable Marriage problem with Incomplete lists and Symmetric Preferences with no ties (SMI-SYM), as shown by the following lemma.

**Lemma 6.3.1.** *Let $J$ be an instance of* SMI-SYM. *Let $M$ be the matching obtained by assigning each man $u$ to the first woman on his preference list. Then, $M$ is the unique popular matching in $J$.*

*Proof.* By symmetry of the preference lists, it is straightforward to see that each woman also obtains her first-choice man in $M$. Hence, $M$ is popular.

We now show that $M$ is unique. For, suppose not. Then, let $M'$ be another popular matching. Clearly, there exists some man $u_i$ such that $M(u_i) \neq M'(u_i)$. Let $M'(u_i) = w_j$ and let $M(u_i) = w_k$. It is straightforward to see that $w_j \neq f(u_i)$ and $u_i \neq f(w_j)$. Now, if $w_j = l(u_i)$, then the matching obtained by $(M' \setminus \{(u_i, w_j), (M'(w_k), w_k)\}) \cup \{(u_i, w_k)\}$ is more popular than $M'$. Hence, $w_j \neq l(u_i)$. Let $M'(w_k) = u_x$. Now, it must be the case that either $u_x \neq l(w_k)$ and $u_x \notin f(w_k)$ or $u_x = l(w_k)$. Let $M(w_j) = u_l$ and let $M'(u_l) = w_y$. Similarly, it must be the case that either $w_y \neq l(u_l)$ and $w_y \notin f(u_l)$ or $w_y = l(u_l)$. Suppose that $u_x$ and $w_y$ are both last resort agents. Then, it is straightforward to verify that the matching obtained by

$$(M' \setminus \{(u_i, w_j), (u_x, w_k), (u_l, w_y)\}) \cup \{(u_i, w_k), (u_l, w_j)\}$$

is more popular than $M'$. Suppose that $a \in \{u_x, w_y\}$ is not a last resort agent. Then, it

is straightforward to verify that the matching obtained by

$$(M' \setminus \{(u_i, w_j), (u_x, w_k), (u_l, w_y)\}) \cup \{(u_i, w_k), (u_l, w_j), (a, l(a))\}$$

is more popular than $M'$. Otherwise, suppose that $u_x$ and $w_y$ are not last resort agents. Then, it is straightforward to verify that the matching obtained by

$$(M' \setminus \{(u_i, w_j), (u_x, w_k), (u_l, w_y)\}) \cup \{(u_i, w_k), (u_l, w_j), (u_x, l(u_x)), (l(w_y), w_y)\}$$

is more popular than $M'$. In all cases, we obtain a contradiction. □

The following lemma is a vital first step in characterising popular matchings in any given SMTI-SYM instance $I$.

**Lemma 6.3.2.** *Let $M$ be a popular matching in $I$. Then $M \cap E_1$ is a maximum matching in $G_1$.*

*Proof.* Let $M_1 = M \cap E_1$. Suppose for a contradiction that $M_1$ is not a maximum matching in $G_1$. Then $M_1$ admits an augmenting path $P = \langle u_1, w_1, ..., u_k, w_k \rangle$ with respect to $G_1$. Clearly, in view of last resort agents, $u_1$ and $w_k$ must be assigned in $M$. Let $w_0 = M(u_1)$ and let $u_{k+1} = M(w_k)$. Now, if both of $w_0$ and $u_{k+1}$ are last resort agents, then it follows that $M \oplus P$ gives us a new matching that is more popular than $M$, a contradiction. On the other hand, if only one of $w_0$ and $u_{k+1}$ is not a last resort agent, then we can demote this agent to his/her last resort partner and use $M \oplus P$ again to obtain a new matching that is more popular than $M$, a contradiction.

Hence, $w_0$ and $u_{k+1}$ are not the last resort partners of $u_1$ and $w_k$ respectively. Furthermore, since $w_0 \notin f(u_1)$ and $u_{k+1} \notin f(w_k)$, it follows that $u_1 \notin f(w_0)$ and $w_k \notin f(u_{k+1})$ respectively. It is clear to see that $w_0 \neq w_i$ $(1 \leq i \leq k-1)$ since each $w_i$ is assigned in $M$ to a first-choice man but $(u_1, w_0) \in M \setminus E_1$. Furthermore, $w_0 \neq w_k$, or else $M' = (M \setminus \{(u_1, w_k)\}) \oplus P$ gives us a more popular matching than $M$, a contradiction. By symmetry of the above argument, it is easy to see that $u_{k+1} \neq u_i$ $(1 \leq i \leq k)$.

Let $w_{k+1}$ be any woman in $f(u_{k+1})$. Then, $u_{k+1} \in f(w_{k+1})$. Suppose firstly that $w_{k+1} = w_0$. Then, it is straightforward to verify that the matching obtained by

$$((M \setminus \{(u_1, w_0), (u_{k+1}, w_k)\}) \oplus P) \cup \{(u_{k+1}, w_0)\}$$

is more popular than $M$, a contradiction. On the other hand, if $w_{k+1} = w_i$ $(1 \leq i \leq k-1)$, let $C = \langle w_i, u_{i+1}, w_{i+1}, ..., u_k, w_k, u_{k+1} \rangle$. It is straightforward to verify that the matching $M' = M \oplus C$ is more popular than $M$, a contradiction. Hence, $w_{k+1} \neq w_i$ $(0 \leq i \leq k)$.

Now, let $u_0 \in f(w_0)$. Then, $w_0 \in f(u_0)$. By symmetry of the above argument, it is straightforward to show that $u_0 \neq u_i$ $(1 \leq i \leq k+1)$.

Now, let $u' = M(w_{k+1})$. If $u' = u_0$, it is straightforward to verify that the matching obtained by

$$((M \setminus \{(u_1, w_0), (u_{k+1}, w_k), (u_0, w_{k+1})\}) \oplus P) \cup \{(u_0, w_0), (u_{k+1}, w_{k+1})\}$$

is more popular than $M$, a contradiction. On the other hand, if $u' = u_i$ $(1 \leq i \leq k)$, then $(u_i, w_{k+1}) \in M$ but since $(u_i, w_{i-1}) \in M$, this implies that $w_{k+1} = w_{i-1}$, which contradicts the fact that $w_{k+1} \neq w_j$ $(0 \leq j \leq k)$ as established above. Hence, the men vertices $u_0, u_1, ..., u_{k+1}, u'$ are all distinct. Let $w' = M(u_0)$. Again, we can reuse the symmetry of the argument to establish that the women vertices $w', w_0, w_1, ..., w_{k+1}$ are all distinct. However, it is straightforward to verify that the matching $M'$ obtained by

$$((M \setminus \{(u_0, w'), (u_1, w_0), (u_{k+1}, w_k), (u', w_{k+1})\}) \oplus P) \cup \{(u_0, w_0), (u_{k+1}, w_{k+1})\}$$

is more popular than $M$ which is a contradiction. (Note that either one of or both $w'$ and $u'$ could possibly be non last resort agents but we can reuse arguments from above to establish that $M'$ is more popular than $M$.) □

Let $I$ be an instance of SMTI-SYM. Clearly, the underlying bipartite graph is uncapacitated. Recall the Edmonds-Gallai Decomposition for the case of an uncapacitated bipartite graph (i.e. Lemma 1.2.1 in Section 1.2). It follows that we can obtain the following corollary by using the Decomposition in conjunction with Lemma 6.3.2.

**Corollary 6.3.1.** *Let $M$ be a popular matching in an instance $I$ of* SMTI-SYM*. Then, every odd or unreachable agent $a \in U \cup W$ satisfies $M(a) \in f(a)$.*

Let $M$ be a popular matching in an instance $I$ of SMTI-SYM. Then, $M_1 = M \cap E_1$ is a maximum matching in $G_1$ by Lemma 6.3.2. Suppose that we are given an EOU labelling of the vertices in $G_1$ using $M_1$. Note that all last resort agents must be even. Now, if an agent $a$ is not assigned to an agent from $f(a)$ in a popular matching $M$, then we will show that $a$ can only be assigned to an agent from his/her set of *s-partner(s)* denoted by $s(a)$, which is a set of agents on $a$'s preference list that is disjoint from $f(a)$. Note that it is easy to see from Corollary 6.3.1 that only even agents should have $s$-partners. We use Algorithm Label-s as shown in Algorithm 11 to define $s(a)$ precisely for every even agent $a$.

**Algorithm 11** Algorithm Label-s

1: $U' := \{u \in U : u \text{ is even}\}$ and $W' := \{w \in W : w \text{ is even}\}$;
2: $E'_i := \{(u, w) \in E : u \in U' \wedge w \in W' \wedge rank_u(w) = i\}$;
3: **for each** $a \in U' \cup W'$ **do**
4: $\quad s(a) := \emptyset$;
5: **for** $i$ in $2..z$ **do**
6: $\quad$ **for each** edge $(u, w) \in E'_i$ **do**
7: $\quad\quad s(u) := s(u) \cup \{w\}$;
8: $\quad\quad s(w) := s(w) \cup \{u\}$;
9: $\quad\quad$ **for each** agent $a \in \{u, w\}$ **do**
10: $\quad\quad\quad$ **for each** agent $b$ after $s(a)$ from $a$'s preference list **do**
11: $\quad\quad\quad\quad k := rank_a(b)$;
12: $\quad\quad\quad\quad$ delete $(a, b)$ from $E'_k$;

The algorithm begins by defining $U'$ and $W'$ to be the respective subsets of $U$ and $W$ containing only even agents. Then, subsets of $E$ are defined, where each subset $E'_i$ contains only $\mathcal{EE}$ edges[2] such that if $(u, w) \in E'_i$, then $u$ and $w$ are both even agents and $rank_u(w) = i$. For each even agent $a$, $s(a)$ is initialised to be the emptyset. The algorithm then iterates over each $i$ in turn from 2 to $z$ and for each value of i, the algorithm iterates over the edges in $E'_i$. Now, if $(u, w)$ is an edge belonging to $E'_i$, then $w$ is added to $s(u)$ and vice versa. As with $f$-partner(s), all $s$-partner(s) of each agent $a$ are tied with the same rank in $a$'s preference list. Hence, when defining $s(a)$, whenever the algorithm has identified a member $c$ of $s(a)$, it will only consider other candidate agents in $a$'s preference list with the same rank as $c$. As a result, if lines 7-8 are executed in iteration $i$, the algorithm removes from consideration certain agents that cannot be a $s$-partner of $a$ in lines 9-12. It does this by deleting $(a, b)$ from $E'_k$ where $rank_a(b) = k$ and $k > i$. Note that lines 9-12 can be executed only once for each edge $e \in E$. Since the number of edges in $G$ are finite, it is clear that the algorithm terminates. When this happens, the $s$-partner(s) of each even agent is defined. Note that $s(a)$ can never become empty for any agent $a$ because of $l(a)$, and in view of the fact that if $a'$ is added to $s(a)$, then this agent cannot be deleted from $s(a)$ subsequently.

Instance $I_5$ in Figure 6.5 gives an illustration of the definition of $f$- and $s$-partners. It is straightforward to verify the $f$-partners for each man and woman. The agents $u_1, u_2, u_3, u_8, w_2, w_5, w_9$ and $w_{10}$ can be verified to be even agents in $G_1$ and their $s$-partners

[2]Recall that these are edges between any two even vertices

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: $w_1$ $w_6$ $w_5$ | $w_1$: ($u_1$ $u_2$) |
| $u_2$: $w_1$ $w_7$ $w_{10}$ $w_2$ | $w_2$: $u_9$ $u_6$ $u_7$ $u_2$ $u_3$ |
| $u_3$: $w_8$ $w_3$ $w_6$ $w_7$ $w_2$ | $w_3$: $u_4$ ($u_3$ $u_8$) |
| $u_4$: ($w_3$ $w_4$) | $w_4$: ($u_4$ $u_5$) |
| $u_5$: $w_4$ ($w_5$ $w_9$) | $w_5$: $u_{10}$ $u_5$ $u_1$ |
| $u_6$: $w_6$ $w_2$ $w_9$ | $w_6$: $u_6$ $u_1$ $u_3$ $u_8$ |
| $u_7$: $w_7$ $w_{10}$ $w_2$ $w_9$ | $w_7$: $u_7$ $u_2$ $u_8$ $u_3$ |
| $u_8$: $w_8$ $w_3$ $w_7$ $w_6$ | $w_8$: ($u_3$ $u_8$) |
| $u_9$: ($w_2$ $w_9$) | $w_9$: $u_9$ $u_5$ $u_6$ $u_7$ |
| $u_{10}$: ($w_5$ $w_{10}$) | $w_{10}$: $u_{10}$ $u_7$ $u_2$ |

Figure 6.5: An instance $I_5$ of SMTI-SYM.

are defined by Algorithm Label-s as follows. In iteration 2, $E'_2 = \emptyset$ so no $s$-partners are defined. In iteration 3, $E'_3$ contains the edges $(u_1, w_5)$ and $(u_2, w_{10})$ so that $s(u_1) = w_5$ and $s(w_5) = u_1$, and $s(u_2) = w_{10}$ and $s(w_{10}) = u_2$. Note that the edges $(u_1, l(u_1))$, $(w_5, l(w_5))$, $(u_2, w_2)$ and $(w_{10}, l(w_{10}))$ are deleted from $E'_4$ by lines 10-12 of the algorithm in this iteration so that $E'_4 = \emptyset$ and no $s$-partners are defined in iteration 4. The edge $(u_2, l(u_2))$ is also deleted from $E'_5$ in iteration 3. In iteration 5, $E'_5$ contains the edges $(u_3, w_2)$, $(u_8, l(u_8))$ and $(w_9, l(w_9))$. Hence, $s(u_3) = w_2$ and $s(w_2) = u_3$, $s(u_8) = l(u_8)$ and $s(w_9) = l(w_9)$. The edges $(u_3, l(u_3))$ and $(w_2, l(w_2))$ are deleted from $E'_6$ by lines 10-12 of the algorithm so that $E'_6 = \emptyset$ and no $s$-partners are defined in the final iteration.

Let $M$ be a popular matching in an instance $I$ of SMTI-SYM. Then, the next two lemmas show that each agent $a$ can only be assigned to a partner from $f(a) \cup s(a)$.

**Lemma 6.3.3.** *Let $M$ be a popular matching in $I$. Then no agent $a \in U \cup W$ can be assigned in $M$ to a partner between $f(a)$ and $s(a)$ on $a$'s preference list.*

*Proof.* Without loss of generality, let us suppose that the agent $a$ is a man $u$; similar results for the women can be obtained by reversing the roles of the sexes in the following proof. Now, suppose that $u$ is assigned to a woman $w$ strictly between $f(u)$ and $s(u)$. If $w$ is odd or unreachable, then by Corollary 6.3.1, $w$ must be assigned to a member of $f(w)$ in $M$, a contradiction. Hence, $w$ is even. By the same argument, $u$ is even. Furthermore, $u$ must be strictly lower than any member of $s(w)$ in $w$'s preference list. For, suppose not. Choose any $u' \in s(w)$. Then, $u$ either (a) precedes $u'$ or (b) is tied with $u'$ in $w$'s

preference list. Let $rank_u(w) = i = rank_w(u)$, and let $rank_w(u') = k$. Now, if case (a) holds, then we have an immediate contradiction since this implies that $i < k$. For, by definition of $s(w)$ according to Algorithm Label-s, there cannot have existed an $\mathcal{EE}$ edge $(u'', w)$ such that $rank_w(u'') = i < k$, or else all members of $s(w)$ would have been deleted from consideration by lines 9-12 of the algorithm in iteration $i$ of the for loop in line 5. On the other hand, if case (b) holds, then $i = k$ so that $(u, w)$ is an edge belonging to $E'_k$, and hence lines 7-8 of Algorithm Label-s would have resulted in $u \in s(w)$ and $w \in s(u)$, a contradiction. Hence, $w$ lies strictly between $f(u)$ and $s(u)$ in $u$'s preference list, and $u$ lies after $s(w)$ in $w$'s preference list. Let us form a new matching $M'$ from $M$ by letting $M' = M \setminus \{(u, w)\}$.

Let $w' \in f(u)$. Since $u$ is even, it follows that $w'$ is odd and hence, it follows by Corollary 6.3.1 that $w'$ is assigned in $M' \cap E_1$ to some $u'$. Clearly, $u' \in f(w')$ and vice versa. Let us now consider $w$ and let $u_0 \in s(w)$. Since $w$ prefers $u_0$ to $u$, it follows that $u_0$ cannot be a last resort agent. Clearly, $u_0$ cannot be $u'$ or else the matching obtained by $(M' \setminus \{(u', w')\}) \cup \{(u, w'), (u', w)\}$ is more popular than $M$, a contradiction. Let $M'(u_0) = w_0$. It follows that $w_0 \neq w'$, for if so then $u_0 = u'$, a contradiction. Furthermore, $w_0$ cannot be worse than $s(u_0)$ in $u_0$'s preference list, or else let $M'' = (M' \setminus \{(u', w'), (u_0, w_0)\}) \cup \{(u, w'), (u_0, w), (u', l(u'))\}$ where we demote $w_0$ to $l(w_0)$ in $M''$ if $w_0 \neq l(u_0)$ (or unassign $w_0$ from $u_0$ otherwise). It follows that $M''$ is more popular than $M$, a contradiction. Hence, it follows that either (i) $w_0$ has the same rank as any member of $s(u_0)$ in $u_0$'s preference list, (ii) $w_0 \in f(u_0)$, or (iii) $w_0$ lies strictly between $f(u_0)$ and $s(u_0)$ in $u_0$'s preference list.

In case (i), let $u_1 \in f(w_0)$. It follows that $w_0$ prefers $u_1$ to $u_0$ since $u_0 \notin f(w_0)$. It is straightforward to verify that $u_1$ cannot be $u$ nor $u'$ or else $M$ cannot be a popular matching. Let $M'(u_1) = w_1$. Then, let $M''$ be the matching obtained by

$$(M' \setminus \{(u', w'), (u_0, w_0), (u_1, w_1)\}) \cup \{(u, w'), (u_0, w), (u_1, w_0), (u', l(u'))\}$$

where we demote $w_1$ to $l(w_1)$ in $M''$ if $w_1 \neq l(u_1)$ (or unassign $w_1$ from $u_1$ otherwise). It follows that $M''$ is more popular than $M$, a contradiction.

In case (ii), since $u_0 \in s(w)$, it is clear that $u_0$ is even and hence, since $w_0 \in f(u_0)$, it follows that $w_0$ is odd. Thus, there exists an odd length alternating path $P' = \langle u_j, w_{j-1}, u_{j-1}, ..., w_1, u_1, w_0 \rangle$ in $G_1$ to $w_0$ from a man $u_j$ who is unassigned in $M' \cap E_1$. Clearly, $u_j \neq u_0$ and $u_j \neq u'$ since these men are assigned to their first-choice women in $M'$. Now, if $u_j = u$, then let $M'' = ((M' \setminus \{(u_0, w_0)\}) \cup \{(u_0, w)\}) \oplus P'$. It is straightfor-

ward to verify that $M''$ is more popular than $M$, a contradiction. Hence, $u_j$ is distinct from all the men agents considered so far. Let $M'(u_j) = w_j$. It follows that $w_j$ is also distinct from all the women agents. Clearly, $w_j$ is worse than any member of $f(u_j)$ in $u_j$'s preference list. Now, if $w_j = l(u_j)$, then the matching obtained by

$$((M' \setminus \{(u', w'), (u_0, w_0), (u_j, w_j)\}) \cup \{(u, w'), (u_0, w), (u', l(u'))\}) \oplus P'$$

can be verified to be more popular than $M$, a contradiction. Hence, $w_j \neq l(u_j)$ and by symmetry, $u_j \notin f(w_j)$. Let $u_{j+1} \in f(w_j)$ instead. Let also $M'(u_{j+1}) = w_{j+1}$. Now, if $u_{j+1}$ is not distinct from any of the above men agents, then we have a cycle $C$. Let $M'' = M' \oplus C$. Then, $M''$ is more popular than $M$, a contradiction. Hence, $u_{j+1}$ (and thus $w_{j+1}$) are distinct agents from those considered so far. However, let $M''$ be the matching obtained by

$$\begin{aligned}&((M' \setminus \{(u', w'), (u_0, w_0), (u_j, w_j), (u_{j+1}, w_{j+1})\}) \oplus P')\\ &\cup \{(u, w'), (u_0, w), (u_{j+1}, w_j), (u', l(u'))\}\end{aligned}$$

where we demote $w_{j+1}$ to $l(w_{j+1})$ in $M''$ if $w_{j+1} \neq l(u_{j+1})$ (or unassign $w_{j+1}$ from $u_{j+1}$ otherwise). It follows that $M''$ is more popular than $M$, a contradiction.

In case (iii), by analogy with $u$ and $w$, it follows that $w_0$ lies strictly between $f(u_0)$ and $s(u_0)$ in $u_0$'s preference list, and $u_0$ lies after $s(w_0)$ in $w_0$'s preference list. Let $u_1 \in s(w_0)$. It follows by a similar argument to the above that $u_1 \neq u'$, or else $M$ cannot be a popular matching. Furthermore, $u_1 \neq u$. For, suppose otherwise. Then, we have the following chain of inequalities:

$$\begin{aligned} rank_w(u) &> rank_w(u_0) = rank_{u_0}(w) \\ &> rank_{u_0}(w_0) = rank_{w_0}(u_0) \\ &> rank_{w_0}(u) = rank_u(w_0) \\ &> rank_u(w) = rank_w(u) \end{aligned}$$

which is a contradiction. It follows that we can reuse arguments from the previous cases to build a path $P = \langle w, u_0, w_0, ..., u_y, w_y \rangle$ starting from $w$. If case (iii) continues to apply, then $P$ will not terminate, a contradiction to the finiteness of the number of agents. Otherwise, either case (i) or (ii) apply and we can obtain a similar contradiction as shown above. □

**Lemma 6.3.4.** *Let $M$ be a popular matching in $I$. Then no agent $a \in U \cup W$ can be assigned in $M$ to a partner worse than $s(a)$ on $a$'s preference list.*

*Proof.* Without loss of generality, suppose that a man $u$ is assigned to a woman $w$ strictly worse than $s(u)$. Clearly, $u \notin f(w)$. By Lemma 6.3.3, $u$ cannot be between $f(w)$ and $s(w)$ on $w$'s preference list. Hence, either (i) $u$ has the same rank as any member of $s(w)$ in $w$'s preference list, or (ii) $u$ is also worse than $s(w)$ on $w$'s preference list. Let us form a new matching $M'$ from $M$ by letting $M' = M \setminus \{(u, w)\}$.

In case (i), let us consider $w$ first. Let $u' \in f(w)$. Clearly, $w$ must be even in $G_1$, for if $w$ is odd or unreachable, then $M(w) \in f(w)$, a contradiction. It follows that $u'$ must be odd in $G_1$, and hence, by Corollary 6.3.1, $u'$ must be assigned in $M'$ to some woman $w' \in f(u')$.

Let $w_0 \in s(u)$. Since $u$ prefers any member of $s(u)$ to $w$, it follows that $w_0 \neq l(u)$. Clearly, $w_0 \neq w'$, for otherwise the matching obtained by $(M' \setminus \{(u', w')\}) \cup \{(u, w'), (u', w)\}$ is more popular than $M$, a contradiction. Let $u_0 = M'(w_0)$. It follows that $u_0 \neq u'$. Now, it must be the case that either (a) $u_0$ is worse that any member of $s(w_0)$ in $w_0$'s preference list, or (b) $u_0$ has the same rank as any member of $s(w_0)$ in $w_0$'s preference list, or (c) $u_0 \in f(w_0)$.

In subcase (a), let $M'' = (M' \setminus \{(u', w'), (u_0, w_0)\}) \cup \{(u', w), (u, w_0), (l(w'), w')\}$ where we demote $u_0$ to $l(u_0)$ in $M''$ if $u_0 \neq l(w_0)$ (or unassign $u_0$ from $w_0$ otherwise). It follows that $M''$ is more popular than $M$, a contradiction.

In subcase (b), let $w_1 \in f(u_0)$. It is straightforward to verify through similar arguments to those used in Lemma 6.3.3 that $w_1 \notin \{w, w'\}$, for otherwise $M$ is not popular. By Corollary 6.3.1, it follows that $u_0$ must be even or otherwise $w_0 \in f(u_0)$, a contradiction. Hence, $w_1$ is odd and $M'(w_1) \in f(w_1)$. Let $M'(w_1) = u_1$. Also, let $M''$ be the matching obtained by

$$(M' \setminus \{(u', w'), (u_0, w_0), (u_1, w_1)\}) \cup \{(u', w), (u, w_0), (u_0, w_1), (u_1, l(u_1)), (l(w'), w')\}$$

It follows that $M''$ is more popular than $M$, a contradiction.

In subcase (c), clearly $w_0$ must be even (as $w_0 \in s(u)$) so that $u_0$ must be odd. Hence, there exists an odd length alternating path $P = \langle w_j, ..., u_0 \rangle$ in $G_1$ from some woman $w_j$ who is unassigned in $M' \cap E_1$ to $u_0$. We can reuse arguments from case (ii) of Lemma 6.3.3 to show that $w_j$ must be a distinct woman from those considered so far. Let $u_j = M'(w_j)$. Now, if $u_j = l(w_j)$, then let $M''$ be the matching obtained by

$$((M' \setminus \{(u', w'), (u_0, w_0), (u_j, w_j)\}) \cup \{(u', w), (u, w_0), (l(w'), w')\}) \oplus P$$

It follows that $M''$ is more popular than $M$, a contradiction. Hence, $u_j \neq l(w_j)$ and clearly,

$w_j \notin f(u_j)$. Hence, let $w_{j+1} \in f(u_j)$ and let $M'(w_{j+1}) = u_{j+1}$. It again follows that we can reuse arguments from case (ii) of Lemma 6.3.3 to show that these agents are distinct from all agents considered so far. However, this implies that we can then obtain a new matching $M''$ by

$$((M' \setminus \{(u', w'), (u_0, w_0), (u_j, w_j), (u_{j+1}, w_{j+1}\}) \oplus P)$$
$$\cup \{(u', w), (u, w_0), (u_j, w_{j+1}), (l(w'), w')\})$$

where we demote $u_{j+1}$ to $l(u_{j+1})$ in $M''$ if $u_{j+1} \neq l(w_{j+1})$ (or unassign $u_{j+1}$ from $w_{j+1}$ otherwise). It follows that $M''$ is more popular than $M$, a contradiction.

In case (ii), it is straightforward to verify that we can reuse the proof for case (i) to show that $M$ must be a popular matching. □

Lemmas 6.3.2-6.3.4 give rise to the following characterisation of popular matchings in any instance $I$ of SMTI-SYM.

**Theorem 6.3.1.** *Let $M$ be a popular matching in any given* SMTI-SYM *instance $I$. Then,*

1. *$M \cap E_1$ is a maximum matching in $G_1$, and*
2. *Every non last-resort agent $a$ is assigned in $M$ to a partner either from $f(a)$ or $s(a)$.*

## 6.4 Structure of popular matchings

Let $G = (U, W, E)$ be the underlying graph of an instance $I$ of SMTI-SYM. We form a subgraph $G'$ of $G$ by letting $G'$ contain only edges from each agent $a \in U \cup W$ to those agents in $f(a) \cup s(a)$. We say that a matching $M$ is *agent-complete* in $G'$ if all those agents that are not last resort agents are assigned in $M$. Clearly, $G'$ need not admit an agent-complete matching if $s(a) \neq \{l(a)\}$ for some agent $a$. It follows by Theorem 6.3.1 that all popular matchings must be contained in $G'$ . However, Theorem 6.3.1 only gives us necessary conditions for a matching to be popular in $I$, since not all matchings in $G'$ satisfying these conditions are popular. For, let us consider the instance $I_5$ in Figure 6.5. Then, we can find at least two matchings which satisfy Conditions 1 and 2 of Theorem

6.3.1, namely

$$M = \left\{ \begin{array}{l} (u_1, w_1), (u_2, w_{10}), (u_3, w_8), (u_4, w_3), (u_5, w_4), \\ (u_6, w_6), (u_7, w_7), (u_9, w_2), (u_{10}, w_5) \end{array} \right\}$$

and

$$M' = \left\{ \begin{array}{l} (u_1, w_5), (u_2, w_1), (u_3, w_2), (u_4, w_3), (u_5, w_4), \\ (u_6, w_6), (u_7, w_7), (u_8, w_8), (u_9, w_9), (u_{10}, w_{10}) \end{array} \right\}$$

However, while $M$ may be verified to be a popular matching as we shall show, $M'$ is not popular because of the cycle $C = \langle u_1, w_1, u_2, w_2, u_3, w_3, u_4, w_4, u_5, w_5 \rangle$. It is straightforward to check that $M' \oplus C$ gives a more popular matching than $M'$. We work towards a necessary and sufficient condition for a matching in SMTI-SYM to be popular in the following subsection.

### 6.4.1 Mutually exclusive edge pairs

Let us define any matching that satisfies Conditions 1 and 2 of Theorem 6.3.1 to be *well-formed*. Clearly, all well-formed matchings must be contained in $G'$. Let $a$ be any even agent in $I$, and let $b$ be any even agent that precedes the members of $s(a)$ in $a$'s preference list. Clearly, $b \notin f(a)$ for otherwise we have an $\mathcal{EE}$ edge in $G_1$, a contradiction by Lemma 1.2.1(c). Then, we define the edge pair $\{(b, b'), (a', a)\}$ to be a *mutually exclusive edge pair*, or *mutex edge pair* for short, if $b' \in f(b)$ and $a' \in s(a)$. The next theorem gives us an important characterisation of popular matchings in SMTI-SYM with respect to mutex edge pairs.

**Theorem 6.4.1.** *Let $M$ be a well-formed matching in any given* SMTI-SYM *instance $I$. Then, $M$ is popular if and only if $M$ does not contain any mutex edge pairs.*

*Proof.* We first show that if $M$ is popular, then $M$ contains no mutex edge pairs. Suppose for a contradiction that $M$ contains the mutex edge pair $\{(u_l, w_j), (u_i, w_p)\}$ such that $u_i$ is an even agent, $w_p \in s(u_i)$ and $w_j$ is an even agent preceding $w_p$ in $u_i$'s preference list and $u_l \in f(w_j)$. We have that $u_i$ prefers $w_j$ to $w_p$ but $w_j$ prefers $u_l$ to $u_i$. Since $w_j$ is an even agent and $w_j$ is assigned in $M$, it follows by the definitions of a well-formed matching and an even vertex that there exists an even length alternating path $P$ in $G_1$ to $w_j$ from an even agent $w_k$ who must be unassigned in $M \cap E_1$, i.e. $M(w_k) \in s(w_k)$. Now, if $w_p = w_k$,

it follows that the sequence of agents $\langle u_i, w_j, ..., w_k \rangle$ then forms a cycle $C$ such that $M \oplus C$ gives us a more popular matching than $M$, a contradiction.

Hence, suppose that $w_k \neq w_p$. Let $M(w_k) = u_a$. Now, if $w_p = l(u_i)$ and $u_a = l(w_k)$, then we can unassign $u_i$ and $w_k$ from their last resort partners, and use $(M \oplus P) \cup \{(u_i, w_j)\}$ to give us a more popular matching than $M$. Let us thus suppose that $w_p = l(u_i)$ and $u_a \neq l(w_k)$. Let $w_b \in f(u_a)$ and let $M(w_b) = u_c$. Now, if $w_b \in P$, then the sequence of agents $\langle w_b, M(w_b), ..., w_k, u_a \rangle$ form a cycle $C$ such that $M \oplus C$ gives us a more popular matching than $M$. Hence, $w_b \notin P$. Since $w_k \in s(u_a)$, it follows that $u_a$ is an even agent so that $w_b$ is odd. By Corollary 6.3.1, it must then be the case that $u_c \neq l(w_b)$. Let $M'$ be the matching obtained by

$$((M \setminus \{(u_i, w_p), (u_a, w_k), (u_c, w_b)\}) \oplus P) \cup \{(u_i, w_j), (u_a, w_b), (u_c, l(u_c))\}$$

It follows that $M'$ is more popular than $M$. On the other hand, if $w_p \neq l(u_i)$ and $u_a = l(w_k)$, then let $u_q \in f(w_p)$ and let $M(u_q) = w_r$. Now, if $u_q \in P$, then the sequence of agents $\langle w_p, u_i, w_j, ..., u_q \rangle$ form a cycle $C$ such that $M \oplus C$ gives us a more popular matching than $M$, a contradiction. Hence, $u_q \notin P$. Reusing a similar argument to the above, we have that $w_r \neq l(u_q)$. However, let $M'$ be the matching obtained by

$$((M \setminus \{(u_q, w_r), (u_i, w_p), (u_a, w_k)\}) \oplus P) \cup \{(u_q, w_p), (u_i, w_j), (l(w_r), w_r)\}$$

It follows that $M'$ is more popular than $M$. Hence, suppose that neither $w_p$ nor $u_a$ is a last resort agent. Then, let $u_q$, $w_r$, $w_b$ and $u_c$ be defined as before. Let $P' = \langle l(w_r), w_r, u_q, w_p, u_i, w_j, ..., w_k, u_a, w_b, u_c, l(u_c) \rangle$. Then, $M \oplus P'$ gives us a more popular matching than $M$. It follows that we obtain a contradiction in all cases so that if $M$ contains a mutex edge pair, then $M$ cannot be popular.

Conversely, let $M$ be a well-formed matching that contains no mutex edge pairs. Suppose for a contradiction that there exists another matching $M' = \{(u_1, w_1), ..., (u_r, w_r)\}$ such that $M'$ is more popular than $M$. We firstly observe that if, for every agent $a_i$ who prefers $M'$ to $M$ $(1 \leq i \leq r)$, his partner in $M'$ prefers $M$ to $M'$, then $M'$ cannot be more popular than $M$. Hence, there exists at least one $a_i$ who prefers $M'$ to $M$ and his partner in $M'$ either (i) also prefers $M'$ to $M$ or (ii) is indifferent between the two matchings. Without loss of generality, let $a_i$ be a man whom we denote by $u_i$ and hence, $M'(u_i) = w_i$ by definition of $M'$. By Theorem 6.3.1 and the definition of a well-formed matching, we can partition the set of agents who are assigned in $M$ into the disjoint sets $F$ and $S$, where agents in $F$ are assigned to their $f$-partners in $M$, and agents in $S$ are assigned to their

$s$-partners in $M$ respectively. It is clear to see that agents in $F$ cannot improve in $M'$ relative to $M$ but can either become worse off or remain indifferent. On the other hand, agents in $S$ can either improve, become worse off in $M'$ relative to $M$ or remain indifferent.

In case (i), it must be the case that each of $u_i$ and $w_i$ can only belong to $S$. It follows that $u_i$ and $w_i$ are both even agents because only even agents have $s$-partners defined. However, this gives a contradiction since Algorithm Label-s would have defined $u_i$ and $w_i$ to be one of each other's $s$-partners because $(u_i, w_i)$ is then an $\mathcal{EE}$ edge such that $u_i$ prefers $w_i$ to any member of $s(u_i)$ and vice versa.

Hence, it remains to consider case (ii). It is clear that $u_i \in S$. Now, if $w_i \in S$ and $w_i$ is indifferent between $u_i$ and $M(w_i)$, we obtain a contradiction as in case (i). Hence, $w_i \in F$. Consider $H' = (M' \oplus M) \cap E_1$. It follows that the only connected components of $H'$ where an agent in $S$ can become assigned to an agent in $F$ who remains indifferent between $M$ and $M'$ are even length alternating paths. Let $u_i$ and $w_i$ belong to such a component $P$. Since $u_i$ improves to $w_i \in F$, and $w_i$ is indifferent between $u_i$ and $M(w_i)$, it follows that $u_i$ is the end vertex of the end edge of $P$ that is in $M'$. It also follows that we have a $u_j$ who is the end vertex of the end edge of $P$ that is in $M$. Clearly, $u_j$ becomes worse off in $M'$ relative to $M$. Now, suppose that $w_j$, who is $u_j$'s partner in $M'$, prefers $M'$ to $M$. By the structure of $P$, $w_j \notin f(u_j)$. Now, if $w_j$ also improves in $M'$ by becoming assigned to $u_j$, it follows that $w_j \in S$ and $u_j$ is an even agent who lies between $f(w_j)$ and $s(w_j)$ in $w_j$'s preference list. However then, it follows that $\{(u_j, M(u_j)), (M(w_j), w_j)\}$ constitutes a mutex edge pair in $M$, a contradiction. Hence, $w_j$ either becomes worse off in $M'$ relative to $M$ or is indifferent between the two matchings. However, it then follows that for every edge $(u_i, w_i)$ where one of the agents improves in $M'$ relative to $M$, exactly one of these agents prefer $M'$ relative to $M$ and the other remains indifferent. Moreover, we have a unique corresponding edge $(u_j, w_j)$ in which at least one of the agents prefers $M$ relative to $M'$ and neither agents prefers $M'$ to $M$. It cannot then be the case that $M'$ is more popular than $M$. □

What Theorem 6.4.1 thus implies is that a well-formed matching $M$ in $G'$ is popular if and only if $M$ contains only one or none of the edges in any mutex edge pair. To illustrate this concept, let us return to instance $I_5$ in Figure 6.5. Let $G'_{I_5}$ be the underlying graph of $I_5$ which contain edges incident to only $f$- and $s$-partners. Then, it may be verified that $G'_{I_5}$ contains one mutex edge pair, namely $\{(u_2, w_1), (u_3, w_2)\}$. Here, $w_2$ and $u_2$ are even agents, $u_2$ precedes $s(w_2) = u_3$ in $w_2$'s preference list, and $w_1 = f(u_2)$. It is thus

straightforward to see that the matching $M$ is popular in $I_5$ because it contains no edges of this mutex edge pair, while $M'$ is not popular because it contains both edges of the mutex edge pair.

### 6.4.2 Testing a matching for popularity

We remark that Theorem 6.4.1 gives us an $O(\sqrt{n}m)$ time algorithm for testing if a given matching $M$ in an SMTI-SYM instance $I$ is popular by checking whether $M$ is a well-formed matching that admits no mutex edge pairs as follows.

First of all, we construct the first-choice graph $G_1$ of $I$ containing only edges incident to $f$-partners in $O(m)$ time. We then find a maximum matching $M_1$ in $G_1$ using the Hopcroft-Karp algorithm in $O(\sqrt{n}m)$ time. We next use $M_1$ to obtain an EOU labelling of the vertices in $G$ in $O(m)$ time through a similar approach outlined in Chapter 4 for the same task in the context of CHAT. It follows that we are then able to identify $\mathcal{E}$, the set of even agents in $I$. Now, the $f$-partners are straightforward to identify. We can then use Algorithm Label-s, as given on page 93, to identify the $s$-partners of each agent in $O(m)$ time (with a suitable choice of data structures such as those described in Section 2.4.3). The search for mutex edge pairs in $M$ can then be done by checking whether the preference list of each even agent $a \in \mathcal{E}$ contains an even agent $b$ preceding any member of $s(a)$ such that $M(b) \in f(b)$ whenever $M(a) \in s(a)$. Clearly, the complexity of this step is bounded by the time required for a traversal of all the preference lists. Hence, we have the following result.

**Lemma 6.4.1.** *Let $M$ be a matching in a given instance of* SMTI-SYM*. Then we may test whether $M$ is popular in $O(\sqrt{n}m)$ time.*

### 6.4.3 Concluding remarks

We conclude with the following observations on mutex edge pairs with respect to popular matchings.

Let $I$ be an instance of SMTI-SYM and let $G'$ be the subgraph of $G$ containing only edges incident to $f$- and $s$-partners constructed as above. Then, it follows by Theorems 6.3.1 and 6.4.1 that the problem of finding a popular matching in $I$, or reporting that none exists, becomes the equivalent problem of finding a well-formed matching in $G'$ that contains no mutex edge pairs, or reporting that none exists.

| Men's pref list | Women's pref list |
|---|---|
| $u_1$: $w_1$ | $w_1$: $u_1$ $u_2$ $u_5$ |
| $u_2$: $w_2$ $(w_1$ $w_4$ $w_5)$ | $w_2$: $u_2$ $u_5$ |
| $u_3$: $w_3$ | $w_3$: $(u_3$ $u_5)$ |
| $u_4$: $(w_4$ $w_5)$ | $w_4$: $u_4$ $u_2$ $u_6$ |
| $u_5$: $w_3$ $w_2$ $w_1$ $w_5$ | $w_5$: $u_4$ $u_2$ $u_6$ $u_5$ |
| $u_6$: $w_6$ $w_7$ $(w_4$ $w_5)$ | $w_6$: $(u_6$ $u_7)$ |
| $u_7$: $w_6$ | $w_7$: $u_8$ $u_6$ |
| $u_8$: $(w_7$ $w_8)$ | $w_8$: $u_8$ |

Figure 6.6: An instance $I_6$ of SMTI-SYM.

Now, a straightforward approach that identifies and deletes all mutex edge pairs from $G'$, and then proceeds to find a well-formed matching in the reduced $G'$ would not work as may be seen from Instance $I_6$ as shown in Figure 6.6. Here, as with instance $I_5$, it is straightforward to identify the $f$-partners. Using Algorithm Label-s, the $s$-partners, where defined, are as follows: $s(u_3) = l(u_3)$, $s(u_5) = w_5$, $s(u_6) = w_7$, $s(u_7) = l(u_7)$, $s(w_4) = l(w_4)$, $s(w_5) = u_5$, $s(w_7) = u_6$, and $s(w_8) = l(w_8)$. It is straightforward to verify that we have two mutex edge pairs as follows:

$$\{(u_6, w_6), (l(w_4), w_4)\} \text{ and } \{(u_6, w_6), (u_5, w_5)\}$$

Let us assume that we delete the above edges from $G'$. Observe that the only edges incident to $w_4$ and $w_5$ in the reduced $G'$ are then $(u_4, w_4)$ and $(u_4, w_5)$. It follows then that no agent-complete matching can exist in the reduced $G'$, causing any such approach to report that no popular matching exists in $I_6$. However, it may be verified that the following is a well-formed matching in $G'$ that contains no mutex edge pairs, and hence is popular in $I_6$ by Theorem 6.4.1:

$$M = \left\{ \begin{array}{l} (u_1, w_1), (u_2, w_2), (u_3, w_3), (u_4, w_4), (u_5, w_5), \\ (u_6, w_7), (u_7, w_6), (u_8, w_8) \end{array} \right\}$$

Another possible solution may be to use a similar approach to that for finding a popular matching in the context of CHA as follows. First, form the subgraph $G_1$ and find a maximum matching $M_1$ of $G_1$. Then, add the edges that are incident to $s$-partners in

$G_1$ to form $G'$, and augment $M_1$ to find an agent-complete matching $M$ of $G'$, if such a matching exists. The objective of these steps are to ensure that $M$ is well-formed. Note that it is straightforward to identify any mutex edge pair(s) once the $f$- and $s$-partners are defined for all agents as described in Section 6.4.2. Let $M_i$ be the matching that is obtained from $M_1$ during a particular iteration $i$ of the augmenting step. We could then try to ensure that $M$ does not contain any mutex edge pair $\{e_1, e_2\}$ by forbidding $M_i$ to be augmented with $e_2$ during iteration $i + 1$ of the augmenting step if $e_1$ already belongs to $M_i$. However, it is unlikely that such a strategy could be successful in general. In fact, the problem of deciding whether there exists an agent-complete matching in a bipartite graph $G'$ without forbidden edge pairs is known to be NP-complete [32].

It therefore remains open as to whether a polynomial-time algorithm can be found to determine whether an instance of SMTI-SYM admits a popular matching. In particular, can we find an efficient way of constructing a well-formed matching without mutex edge pairs, if such a matching exists? If such an algorithm can be found, then it could form the basis of an approach to solve the analogous problem in the general SMTI and SRTI cases.

# Chapter 7

# Profile-based optimal matchings in CHAT

## 7.1 Introduction

As discussed in Section 1.3.2, given any matching $M$ in a bipartite matching problem, various optimality criteria based on the profile of $M$ may be used to determine the quality of $M$ with respect to other matchings in the same problem instance. In this chapter, we study several optimality concepts for bipartite matching problems based on the profile of matchings in the context of CHAT. The three optimality criteria that we study in this chapter are the notions of a *greedy maximum* matching, a *rank-maximal* matching, and a *generous maximum* matching, as introduced in Section 1.3.2. We remark that these concepts are particularly useful in many practical matching applications where the foremost goal of the matching scheme is to maximise the number of participating agents who are assigned, and then subject to this constraint, to optimise the satisfaction of the agents with respect to their preferences.

The main results of this chapter, and their organisation are as follows. First of all, Section 7.2 introduces the terminology and notations that will be used for the rest of this chapter. Next, Section 7.3 presents an $O(C^2mz)$ time algorithm (based on a variant of the Bellman-Ford algorithm [28]) to find a greedy maximum matching given an CHAT instance, where $C$ is the total capacity of houses, $m$ is the total length of preference lists and $z$ is the maximum rank respectively in the problem instance. Section 7.4 presents an $O(\min(z^*\sqrt{C}, C + z^*)m)$ time algorithm that uses the Edmonds-Gallai Decomposition to find a rank-maximal matching given an CHAT instance, where $z^*$ is the maximal

rank in an optimal solution. In that section, we also present a number of straightforward alternative algorithms to solve the problem and show how our direct approach based on the Edmonds-Gallai Decomposition is faster than each of these. Finally, Section 7.5 shows how the algorithms for finding a greedy maximum matching for a given CHAT instance can be adapted for finding a generous maximum matching in the same problem instance.

## 7.2 Basic terminology

Let $I$ be an instance of CHAT as defined in Chapters 1, 2 and 4. Let $z$ be the maximum rank of a house taken over all agents' preference lists in $I$. Let $\mathcal{M}$ be the set of all matchings of $A$ to $H$. The following definition gives a property of matchings given an instance of CHAT.

**Definition 7.2.1.** *The profile $\rho(M)$ of a matching $M \in \mathcal{M}$ is defined to be the $z$-tuple $(x_1, x_2, ..., x_z)$ where for each $i$ $(1 \leq i \leq z)$, $x_i$ is the number of agents who are assigned in $M$ with one of their $i$th choice houses.*

For a given CHAT instance $I$, a *feasible $s$-profile* is a profile $X = (x_1, ..., x_z)$ such that there is a matching $M$ for $I$ with profile $X$ where $|M| = s$. It is immediate that $\sum x_i = s$. To simplify matters, we abbreviate a profile $(x_1, ..., x_z)$ by $(x_1, ..., x_d)$ if $x_d > 0$ and $x_i = 0$ for $i = d+1, ..., z$. We let the empty matching have profile (0). We may define a total order $\succ_L$ on profiles as follows: let $Y = (y_1, ..., y_z)$ and $X = (x_1, ..., x_z)$ be any two profiles. Then, $Y \succ_L X$ if there exists some $k$ $(1 \leq k \leq z)$ such that $x_i = y_i$ for $1 \leq i < k$ and $y_k > x_k$. We say that $y$ *left-dominates* $x$. Let $O = (o_1, ..., o_z)$ be the $z$-tuple such that $o_i = 0$ for $1 \leq i \leq z$. It follows that the profile of any non-empty matching must left-dominate $O$.

Alternatively, we may define a second total order $\prec_R$ on profiles as follows: $X \prec_R Y$ if there exists some $k$ $(1 \leq k \leq z)$ such that $x_i = y_i$ for $k < i \leq z$ and $x_k < y_k$. We say that the profile $X$ *right-dominates* profile $Y$. Let $O' = (o'_1, ..., o'_z)$ be the $z$-tuple such that $o'_i = 0$ for $1 \leq i \leq z-1$ and $o'_z = C+1$. Then, it follows that the profile of any non-empty matching must right-dominate $O'$. It is straightforward to see that each of $\succ_L$ and $\prec_R$ is transitive. Let $G$ be the underlying graph of $I$. Then, we define the profile of a connected component $C$ in $G$ with respect to a matching $M$ in $G$ to be the $z$-tuple $\rho_C(M) = (\alpha_1, ..., \alpha_z)$ where for each $i$ $(1 \leq i \leq z)$, $\alpha_i$ is the number of agents in $C$ who obtain their $i$th-choice house (in $I$) in $M$. Given any two profiles $\rho_1 = (\beta_1, ..., \beta_z)$ and

**Algorithm 12** Algorithm Greedy-Max

```
1: M := ∅;
2: s := 0; {// s is the cardinality of M}
3: loop
4:    P := a maximum profile augmenting path for M;
5:    if P exists then
6:       M := M ⊕ P;
7:    else
8:       exit;
9:    s := s + 1;
10: return M; {// a greedy maximum matching}
```

$\rho_2 = (\gamma_1, ...\gamma_z)$, we define the sum of $\rho_1 + \rho_2$ to be $(\beta_1 + \gamma_1, ..., \beta_z + \gamma_z)$. Furthermore, $\rho_1 = \rho_2$ if $\beta_i = \gamma_i$ for all $i$ $(1 \leq i \leq z)$.

## 7.3 Greedy maximum matchings

For a given instance $I$ of CHAT, we say that a feasible $s$-profile $X$ is *$s$-left-maximal* if there is no other $s$-profile that left-dominates $X$. We define a matching $M$ whose profile is $s$-left-maximal to be a *greedy* $s$-matching. When $s$ is the cardinality of a maximum matching, we say that a greedy $s$-matching is a *greedy maximum* matching. Note that there may be more than one greedy $s$-matching for any value of $s$, but it must be the case that all greedy $s$-matchings have the same profile, and we call this the *greedy $s$-profile* for the problem instance. When $s$ is the cardinality of a maximum matching, we call this the *greedy maximum profile*. Let $\mathcal{M}^+$ denote the set of maximum matchings in $\mathcal{M}$. Then, we may formalise the definition of a greedy maximum matching in a CHAT instance as follows.

**Definition 7.3.1.** *Given an instance of* CHAT*, a greedy maximum matching is a maximum matching that has maximum profile under the order $\succ_L$ taken over all matchings in $\mathcal{M}^+$.*

### 7.3.1 Finding a greedy maximum matching

We now introduce our algorithm to find a greedy maximum matching in any given CHAT instance $I$. Our algorithm extends an existing approach for finding a greedy maximum matching, given an instance of the House Allocation problem with Ties, and is

| Agent | Pref list | House | Capacity |
|---|---|---|---|
| $a_1$: | $h_1$ $h_2$ | $h_1$ : | 1 |
| $a_2$: | $h_1$ $h_2$ | $h_2$ : | 1 |
| $a_3$: | $h_4$ $h_3$ | $h_3$ : | 1 |
| $a_4$: | $h_4$ | $h_4$ : | 1 |

Figure 7.1: An instance $I_1$ of CHAT

based on a variant of the Bellman-Ford algorithm [28]. A pseudocode description of the main loop of the algorithm is given in Algorithm 12. This adapts the classical augmenting path algorithm for finding a maximum matching in a graph. That is, we start from the empty matching, and then repeatedly increase the cardinality of the current matching via an augmenting path until no such path can be found. In our algorithm, however, we aim to satisfy the greedy maximum condition by looking for greedy $s$-matchings, and augmenting the current (greedy) matching at any stage of the algorithm by using only an augmenting path which leads to a greedy $(s+1)$ matching, provided $s$ is not the cardinality of a maximum matching.

We would like to show that it always suffices to use a single augmenting path at each stage $s$ of the algorithm to obtain a greedy $(s+1)$-matching from a greedy $s$-matching. For instance, consider the CHAT example in Figure 7.1. Then, the matching $M_1 = \{(a_1, h_1), (a_2, h_2), (a_3, h_4)\}$ is a greedy 3-matching. Now, the matchings $M_2 = \{(a_1, h_2), (a_2, h_1), (a_3, h_3), (a_4, h_4)\}$ and $M_3 = \{(a_1, h_1), (a_2, h_2), (a_3, h_3), (a_4, h_4)\}$ are two possible greedy 4-matchings that could be obtained from $M_1$. However, while it takes an augmenting path and an alternating cycle to move from $M_1$ to $M_2$, it takes only an augmenting path to move from $M_1$ to $M_3$. The next lemma proves that a single augmenting path always suffices to obtain a greedy $(s+1)$-matching from a greedy $s$-matching.

**Lemma 7.3.1.** *Let $M$ be a greedy $s$-matching in $I$. Then either $M$ is a greedy maximum matching or there is a greedy $(s+1)$-matching $M'$ that can be obtained from $M$ via an augmenting path.*

*Proof.* Let $G$ be the underlying bipartite graph of $I$. Suppose that $M$ is not a greedy maximum matching. Hence, there exists a greedy $(s+1)$-matching $M_1$. We clone $G$ to obtain a cloned graph $C(G)$ as follows. We replace every house $h_j \in H$ with the clones $h_j^1, h_j^2, \ldots, h_j^{c_j}$. We then divide the capacity of each house among its clones by allowing each clone to have capacity 1. In addition, if $(a_i, h_j) \in E$, then we add $(a_i, h_j^p)$ to the

edge set of $C(G)$ for all $p$ $(1 \leq p \leq c_j)$. Furthermore, if $rank_{a_i}(h_j) = k$, then we let $rank_{a_i}(h_j^p) = k$ for all $p$ $(1 \leq p \leq c_j)$, so that $h_j^p$ is a $k$th-choice house for $a_i$ in $C(G)$ for all $p$ $(1 \leq p \leq c_j)$. Let us then adapt the matching $M$ in $G$ to obtain its clone $C(M)$ in $C(G)$ as follows. If a house $h_j$ in $G$ is assigned to $x_j$ agents $a_1, ...a_{x_j}$ in $M$, then we add $(a_p, h_j^p)$ to $C(M)$ for $1 \leq p \leq x_j$, so that $|C(M)| = |M|$. We use a similar process for $M_1$ to obtain its clone $C(M_1)$ in $C(G)$. Hence, $C(M)$ is a greedy $s$-matching and $C(M_1)$ is a greedy $(s+1)$-matching in $C(G)$.

Let us consider $X = C(M) \oplus C(M_1)$. Then, it follows that each connected component of $X$ is either (i) an alternating cycle, (ii) an even length alternating path, or (iii) an odd length alternating path. We will show that there exists a greedy $(s+1)$-matching $M'$ such that we require only a connected component of type (iii) in order to obtain $M'$ from $M$.

Let $D$ be a connected component of $X$ that is either (i) or (ii). Let $\rho_D(C(M)) = (a_1, ..., a_z)$, and let $\rho_D(C(M_1)) = (b_1, ..., b_z)$. Suppose that $\rho_D(C(M_1)) \succ_L \rho_D(C(M))$. Then, we can create a new matching $C_1$ of cardinality $s$ in $C(G)$ by replacing the $C(M)$-edges in $D$ by the $C(M_1)$-edges in $D$, giving $\rho(C_1) \succ_L \rho(C(M))$, which is a contradiction since $C(M)$ is a greedy $s$-matching in $C(G)$. A similar contradiction arises if $\rho_D(C(M)) \succ_L \rho_D(C(M_1))$. Hence, $\rho_D(C(M)) = \rho_D(C(M_1))$.

Now, let us form another greedy $(s+1)$-matching $C_2$ in $C(G)$ from $C(M_1)$ by replacing every $C(M_1)$-edge by the corresponding $C(M)$-edge in each connected component. Consider each connected component $F$ of $Y = C(M) \oplus C_2$. Then, it follows that $F$ can only be an odd length alternating path. By the existence of $C_2$, there must exist an odd number of such paths in $Y$ in order for us to be able to augment $C(M)$ to $C_2$. Now, if only one such path $P'$ exists, then it follows that $P'$ is an augmenting path and $C(M) \oplus P'$ gives us the greedy $(s+1)$-matching $C_2$.

Otherwise, there is more than one such path. Let $P_1$ and $P_2$ be any two such paths which together have the same number of $C(M)$- and $C_2$-edges; it must be possible to find such a pair of paths. Let $\rho_{P_1}(C(M)) + \rho_{P_2}(C(M)) = (\alpha_1, ..., \alpha_z)$ and $\rho_{P_1}(C_2) + \rho_{P_2}(C_2) = (\beta_1, ..., \beta_z)$. Suppose that $(\beta_1, ..., \beta_z) \succ_L (\alpha_1, ..., \alpha_z)$. Then, we can create a new matching $C_3$ of cardinality $s$ in $C(G)$ by replacing the $C(M)$-edges by the $C_2$-edges in $P_1$ and $P_2$ respectively, giving $\rho(C_3) \succ_L \rho(C(M))$, which contradicts the fact that $C(M)$ is a greedy $s$-matching in $C(G)$. We obtain a similar contradiction if $(\alpha_1, ..., \alpha_z) \succ_L (\beta_1, ..., \beta_z)$. Hence, $\rho_{P_1}(C(M)) + \rho_{P_2}(C(M)) = \rho_{P_1}(C_2) + \rho_{P_2}(C_2)$. Let us form another greedy $(s+1)$-matching $C_4$ in $C(G)$ from $C_2$ by replacing every $C_2$-edge by the corresponding $C(M)$-edge in every

such pair of odd length alternating paths. Then, it follows that $C(M) \oplus C_4$ contains only a single alternating path $P'$ of odd length. Moreover, $P'$ is an augmenting path and $C(M) \oplus P'$ gives us the greedy $(s+1)$-matching $C_4$.

Consider the path $P'$. Let $C' = C(M) \oplus P'$. Replace each cloned edge $(a_i, h_j^p)$ in $P'$ by the original edge $(a_i, h_j)$ in $G$ from which it was derived, where $1 \leq p \leq c_j$. It follows that $P'$ becomes an augmenting path $P$ with respect to $M$. Perform the same edge replacements for $C'$ to obtain a matching $M'$ in $G$. It follows that $M \oplus P = M'$. Since $\rho(M') = \rho(C')$, it must be the case that $M'$ is a greedy $(s+1)$-matching in $G$. Hence, the result follows. □

Given any greedy $s$-matching $M$ in $I$ that is not a greedy maximum matching, the task at hand now is to be able to identify an augmenting path which will lead us to a greedy $(s+1)$-matching. To do so, let us introduce the notion of a *maximum profile augmenting path*. Let $\alpha$ be an integer such that $1 \leq \alpha \leq z$. We define $X + \alpha$ to be

$$X + \alpha = (x_1, ..., x_{\alpha-1}, x_\alpha + 1, x_{\alpha+1}, ..., x_z)$$

and we define $X - \alpha$ to be

$$X - \alpha = (x_1, ..., x_{\alpha-1}, x_\alpha - 1, x_{\alpha+1}, ..., x_z)$$

Let $P = \langle a_0, h_0, a_1, h_1, ..., a_x, h_x \rangle$ be an alternating path from an exposed agent vertex $a_0$ to a house vertex $h_x$, such that $(a_i, h_{i-1}) \in M$ for $1 \leq i \leq x$. We then define the *profile* of $P$ to be

$$\begin{aligned} \rho(P) &= O + r(a_0, h_0) + r(a_1, h_1) + ... + r(a_x, h_x) \\ &\quad - r(a_1, h_0) - r(a_2, h_1) - ... - r(a_x, h_{x-1}) \end{aligned}$$

It follows that if $P$ is an augmenting path, then $\rho(P)$ corresponds to the net change in the profile of $M$ if we augment $M$ along $P$. For every house vertex $h_j \in H$, we define the *L-value* of $h_j$ relative to $M$, denoted by $L(h_j)$, to be the maximum profile taken over all alternating paths from an exposed agent vertex ending at $h_j$, where a vertex is defined to be exposed if it is unmatched in $M$. We say that an alternating path $P$ is a *maximum profile* augmenting path for $M$ if $P$ is an augmenting path, and $\rho(P) = \max\{L(h_j) : h_j \in H\}$ where max is with respect to the $\succ_L$ order on profiles.

The following lemma shows that we can use the notion of a maximum profile augmenting path in tandem with the classical augmenting path algorithm to find a greedy maximum matching in CHAT.

**Algorithm 13** Algorithm Max-Aug

```
1: Initialise l(h_j) and pred(h_j) for each house vertex h_j;
2: for p in 1..s do
3:    for each agent vertex a_i assigned in M do { // M is the current greedy s-matching}
4:       for each edge (a_i, h_j) ∉ M do
5:          σ := l(M(a_i)) + r(a_i, h_j) − r(a_i, M(a_i));
6:          if σ ≻_L l(h_j) then
7:             l(h_j) := σ;
8:             pred(h_j) := a_i;
```

**Lemma 7.3.2.** *Suppose that $M$ is a greedy $s$-matching which is not maximum. Let $P$ be a maximum profile augmenting path. Then, augmenting $M$ along $P$ gives a greedy $(s+1)$-matching.*

*Proof.* Suppose for a contradiction that $M' = M \oplus P$ does not give us a greedy $(s+1)$-matching. Now, by Lemma 7.3.1, there exists a matching $M''$ of cardinality $s+1$ such that $M'' \succ_L M'$ and $M'' = M \oplus P'$ for an augmenting path $P'$. Since $M'' \succ_L M'$, it follows that $M \oplus P' \succ_L M \oplus P$, i.e. $P' \succ_L P$. However, this gives a contradiction since $P$ is an augmenting path of maximum profile for $M$. □

Let $M$ be any greedy $s$-matching that is not a greedy maximum matching for a given CHAT instance $I$. It now remains to show how to find a maximum profile augmenting path with respect to $M$. We do this using the algorithm shown in Algorithm 13, which is a variant of the Bellman-Ford algorithm for finding shortest paths. In the algorithm, we will use $l(h_j)$ as described below to compute $L(h_j)$ for each house vertex $h_j$. We will also use a predecessor value $pred(h_j)$ to store the agent vertex preceding $h_j$ in the alternating path which has maximum profile among all alternating paths from an exposed agent vertex to $h_j$ found so far by the algorithm. At the start of the algorithm, we initialise $l(h_j)$ and $pred(h_j)$ for each house vertex $h_j$ as follows. If there is an edge $(a_i, h_j)$ incident to $h_j$ such that $a_i$ is currently exposed, let $t$ be the minimum value of $r(a_i, h_j)$ taken over all such edges, where recall that $r(a_i, h_j) = rank_{a_i}(h_j)$. We then initialise $l(h_j)$ to be the $t$-tuple $(x_1, ..., x_t)$ where $x_p = 0$ $(1 \leq p < t)$ and $x_t = 1$. Furthermore, we initialise $pred(h_j)$ to be $a_i$. If no such edge exists, we initialise $l(h_j)$ to be (0) and $pred(h_j)$ is undefined. Intuitively, $l(h_j)$ gives the maximum profile of any alternating path of length 1 from an exposed agent vertex $a_i$ to $h_j$ at the start of the algorithm, and so $pred(h_j)$ is set to be $a_i$, if such a path exists.

The algorithm runs in $s$ iterations, where $s$ is the cardinality of the current matching as constructed by Algorithm Greedy-Max (see Algorithm 12). It uses an edge relaxation operation similar to that of the Bellman-Ford algorithm, but bases this operation in terms of the order $\succ_L$ on $L$-values. The edge relaxation operation is defined in line 5 of the algorithm. Let $a_i$ be any agent vertex assigned in the current greedy $s$-matching $M$ with $(a_i, h_j) \notin M$. Also, let $P = \langle v, ..., M(a_i) \rangle$ be an alternating path, starting from an exposed agent vertex $v$ and ending at $M(a_i)$, whose profile is equal to $l(M(a_i))$. The essence of the edge relaxation operation is the following: if the profile of the alternating path $P' = \langle v, ..., M(a_i), a_i, h_j \rangle$ left-dominates $l(h_j)$, i.e. $P'$ gives a "better profile" than the alternating path whose profile is equal to $l(h_j)$, then we update $l(h_j)$ to be the profile of $P'$ and similarly update the predecessor of $h_j$ to be $a_i$.

Now, if $l(h) = (0)$ for every house vertex $h$ after execution of Algorithm Max-Aug, then there is no augmenting path, and $M$ is a greedy maximum matching. Otherwise, we find an exposed house vertex $h_j$ such that $l(h_j)$ left-dominates the $L$-values of all exposed house vertices. The correctness proof in the next section shows that we can obtain the maximum profile augmenting path $P$ by alternately tracing the predecessor values and matched edges starting from $pred(h_j)$.

### 7.3.2 Proof of correctness

Let $Y = (y_1, ..., y_z)$ and $X = (x_1, ..., x_z)$ be any two profiles. We introduce a new notation as follows, that is, $Y \succeq_L X$ if there exists some $k$ $(1 \leq k \leq z)$ such that $x_i = y_i$ for $1 \leq i < k$ and $y_k \geq x_k$. Intuitively, this implies that either $Y = X$ or $Y \succ_L X$. The following lemma shows us that Algorithm Max-Aug correctly computes the $L$-value of each house vertex.

**Lemma 7.3.3.** *When Algorithm Max-Aug terminates, $l(h_j) = L(h_j)$ for each house vertex $h_j \in H$.*

*Proof.* Let $h_j$ be an arbitrary house vertex in $G$. Let also $L_{2p+1}(h_j)$ denote the maximum profile of any alternating path of length $\leq 2p + 1$ from an exposed agent vertex to $h_j$. We will prove the following loop invariant: after iteration $p$ of the main for loop, $l(h_j)$ is equal to or left-dominates the maximum profile taken over all alternating paths of length $\leq 2p + 1$ from an exposed agent vertex to $h_j$, i.e. $l(h_j) \succeq_L L_{2p+1}(h_j)$.

For the base case, let $p = 0$. Now, $L_1(h_j)$ is the maximum profile of any alternating path of length $\leq 1$ from an exposed agent vertex to $h_j$. This is precisely the value that

$l(h_j)$ is initialised to in line 1 of the algorithm as discussed on the previous page. Hence, the base case holds.

For the induction step, we assume that $1 \leq p \leq s$ and that the loop invariant is true for $p-1$, i.e. $l_{prev}(h) \succeq_L L_{2p-1}(h)$ after the $(p-1)$th iteration for all $h \in H$, where $l_{prev}(h)$ denotes the value of $l(h)$ as computed after the $(p-1)$th iteration. We will show that the loop invariant holds after iteration $p$, that is $l(h_j) \succeq_L L_{2p+1}(h_j)$ where $l(h_j)$ is computed after the $p$th iteration. Let $A'_j = \{a_i \in A : (a_i, h_j) \in E \backslash M$ and $a_i$ is assigned in $M\}$ where $E$ is the edge set in $G$. During each iteration, we perform a relaxation step for every edge $(a_i, h_j)$ such that $a_i \in A'_j$.

By definition, $L_{2p+1}(h_j)$ denotes the maximum profile of any alternating path of length $\leq 2p+1$ from an exposed agent vertex to $h_j$. Now, if (i) there does not exist any alternating path from an exposed agent vertex of length $\leq 2p+1$ which gives a better profile than $L_{2p-1}(h_j)$, then it follows that $L_{2p+1}(h_j) = L_{2p-1}(h_j)$. Otherwise, it must be the case that (ii) an alternating path of length $\leq 2p+1$ from an exposed agent vertex to $h_j$ with profile $L_{2p+1}(h_j)$ must contain an alternating path from an exposed agent vertex to $h'_j$ with profile $L_{2p-1}(h'_j)$ for some house vertex $h'_j$, together with the edges $(M(h'_j), h_j)$ and $(M(h'_j), h'_j)$. In such a case, it follows that

$$L_{2p+1}(h_j) = \max\left\{L_{2p-1}(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}$$

Now, by inspection of the algorithm, we have that

$$l(h_j) = \max\left\{l_{prev}(h_j), \left\{l'(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}\right\}$$

where $l'(M(a_i))$ is the most recent $L$-value of $M(a_i)$ when $l(h_j)$ is updated by the algorithm; it must be the case that $l_{prev}(M(a_i)) \preceq_L l'(M(a_i))$. By definition of $L_{2p+1}(h_j)$, it follows that

$L_{2p+1}(h_j)$

$$\begin{aligned}
&= \max\left\{L_{2p-1}(h_j), \max\left\{L_{2p-1}(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}\right\} \\
&\preceq_L \max\left\{l_{prev}(h_j), \max\left\{L_{2p-1}(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}\right\} \\
&\quad \text{(by the inductive hypothesis)} \\
&\preceq_L \max\left\{l_{prev}(h_j), \max\left\{l_{prev}(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}\right\} \\
&\quad \text{(by the inductive hypothesis)} \\
&\preceq_L \max\left\{l_{prev}(h_j), \max\left\{l'(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i)) : a_i \in A'_j\right\}\right\} \\
&= l(h_j)
\end{aligned}$$

completing the inductive step.

Thus, after iteration $s$, we have that $l(h_j) \succeq_L L_{2s+1}(h_j)$. However, any alternating path from an exposed agent vertex to $h_j$ with respect to the greedy $s$-matching can have length at most $2s+1$. Hence, we have that $l(h_j) = L(h_j)$ after iteration $s$, i.e. $l(h_j)$ is equal to the maximum profile taken over all alternating paths from an exposed agent vertex to $h_j$ as required. □

Let $M$ be a greedy $s$-matching that is not a maximum matching. Let $h_r$ be an exposed house vertex whose $L$-value left-dominates the $L-$values of all exposed house vertices after execution of Algorithm Max-Aug. Let $P$ be the sequence of agents and houses obtained by alternately tracing the predecessor values and matched edges in $M$ starting from $h_r$. The next lemma shows that $P$ must terminate at some exposed agent vertex.

**Lemma 7.3.4.** *Let $P$ be the sequence of agents and houses obtained by alternately tracing the predecessor values and matched edges in $M$ starting from $h_r$ where $h_r$ is defined as above. Then, $P$ terminates at some exposed agent vertex.*

*Proof.* We wish to show that $P$ cannot cycle, so let us suppose the contrary for a contradiction, i.e. $P$ contains some cycle $C$. Hence, there must have been some point during the execution of Algorithm Max-Aug when $C$ appeared for the first time. Call this step $X$. Suppose that this happened when some house $h_j \in P$ had its predecessor assigned to the agent $a_i \in P$.

We firstly observe that $h_j$ must itself be in $C$, for otherwise, since none of the other house vertices had its predecessor changed at step $X$, and none of the agents had his assigned house changed at step $X$, $C$ must have existed before step $X$, a contradiction to the fact that $C$ appears for the first time. It follows that we can trace an alternating path from each of the other house vertices $h$ in $C$ to $h_j$ by following the predecessor values and matched edges in $M$ from $pred(h)$. Hence, immediately prior to step $X$, $h_j$ must itself have had a defined predecessor, and an existing value for $l(h_j)$.

Since $pred(h_j)$ was assigned to $a_i$, it must have been the case that $\sigma \succ_L l(h_j)$, where $l(h_j)$ was the existing $L$-value for $h_j$ and $\sigma = l(M(a_i)) + r(a_i, h_j) - r(a_i, M(a_i))$. However, it is also the case that $\sigma = l(h_j) - \rho_C(M) + \rho_C(M')$ where we let $M' = M \oplus C$. Hence, we have that $\rho_C(M') \succ_L \rho_C(M)$. However, this implies that $M'$ is a matching such that $|M'| = |M|$ and $M' \succ_L M$, a contradiction. □

By Lemma 7.3.4, it follows that we can use the predecessor values to successfully trace

an augmenting path $P$ with respect to $M$ where $P$ ends at the house vertex $h_r$. By Lemma 7.3.3 and the definition of $L(h_r)$, $P$ must be a maximum profile augmenting path with respect to $M$ if $l(h_r)$ left-dominates the $L$-values over all exposed house vertices. This gives us the following result.

**Theorem 7.3.1.** *Let $M$ be a greedy $s$-matching that is not greedy maximum. Then, executing Algorithm Max-Aug finds a maximum profile augmenting path with respect to $M$.*

### 7.3.3 Time complexity analysis

The time complexity for finding a greedy maximum matching in an instance $I$ of CHAT may be derived as follows.

Algorithm Greedy-Max performs $S$ calls to Algorithm Max-Aug to find maximum profile augmenting paths where $S$ is the cardinality of a maximum matching in $I$. The main for loop of Algorithm Max-Aug itself is performed $O(s)$ times, where $s$ is the cardinality of the current matching. Each iteration of the innermost for loop in line 6 makes a comparison of $l$-values which takes $O(z)$ time. There can be $O(m)$ such iterations altogether during a single execution of Algorithm Max-Aug. At the end of Algorithm Max-Aug, $O(n_2)$ comparisons of $L$-values, each of which takes $O(z)$ time, are made to identify a maximum profile augmenting path. Hence, each execution of Algorithm Max-Aug takes $O(smz + n_2z) = O(smz)$ time. It follows that the overall time complexity of Algorithm Greedy-Max is $O(C^2mz)$ since the maximum cardinality of a matching in an instance of CHAT is $O(C)$. In practice, the actual runtime of Algorithm Greedy-Max can be speeded up through the observation that if no house $h$ that is assigned in the current matching had $l(h)$ updated in the last iteration of Algorithm Max-Aug, then no further improvement to an $l(h')$ for all $h' \in H$ can happen. Hence, we may choose to halt Algorithm Max-Aug at that point.

## 7.4 Rank-maximal matchings

Let $I$ be a given instance of CHAT. We formalise the definition of a rank-maximal matching in $I$ as follows.

**Definition 7.4.1.** *Given an instance of* CHAT*, a rank-maximal matching is a matching that has maximum profile under the order $\succ_L$, taken over all matchings in $\mathcal{M}$.*

| Agent | Pref list | House | Capacity |
|---|---|---|---|
| $a_1$: | $h_1$ $h_2$ | $h_1$ : | 1 |
| $a_2$: | $h_1$ | $h_2$ : | 2 |

Figure 7.2: An instance $I_2$ of CHAT

For a given CHAT instance, there may be more than one rank-maximal matching, but all rank-maximal matchings must have the same profile, and hence the same cardinality. Now, it is straightforward to see that a simple greedy algorithm in which we assign the maximum number of agents to their first choice house, then the maximum number to their second choice house, and so on, does not guarantee to find a rank-maximal matching. For example, Figure 7.2 shows a given CHAT instance $I_2$ in which the greedy algorithm may either return matching $M_1 = \{(a_1, h_1)\}$ given the agent ordering $\langle a_1, a_2 \rangle$, or $M_2 = \{(a_1, h_2), (a_2, h_1)\}$ given the agent ordering $\langle a_2, a_1 \rangle$, each of which maximises the number of agents assigned to their first choice house. However, $M_2$ is rank-maximal but $M_1$ is not, since $\rho(M_1) = (1)$, but $\rho(M_2) = (1, 1)$.

### 7.4.1 Finding a rank-maximal matching

For the special case where every house has unitary capacity, i.e. the House Allocation problem with Ties (HAT), Irving et al. [29] give an $O(\min(z^*\sqrt{n}, n + z^*)m)$ (direct) combinatorial algorithm for solving the analogous problem, where $z^* \leq z$ is the maximal rank of an edge used in a rank-maximal matching. We will show how to extend this algorithm to the CHAT case.

Let $G$ be the underlying graph of $I$. Let $E_i$ denote the set of edges having rank $i$ for any $i$ $(1 \leq i \leq z)$. Then, the edge set of $G$ may be expressed as $E = E_1 \cup E_2 \cup ... \cup E_z$. Our algorithm works in phases. In each phase $i$ $(1 \leq i < z)$, it constructs a rank-maximal matching $M_{i+1}$ of the subgraph $G_{i+1} = (A \cup H, E_1 \cup E_2 \cup ... \cup E_i \cup E_{i+1})$ so that when it terminates at the end of phase $z - 1$, a rank-maximal matching $M_z$ of the subgraph $G_z$ is a rank-maximal matching for $I$. Our algorithm begins at the outset by constructing a maximum matching $M_1$ in $G_1 = (A \cup H, E_1)$. Note that $M_1$ is a rank-maximal matching of $G_1$ since it is a maximum matching of $E_1$ edges. For each phase $i$ where $i \geq 1$, our algorithm then constructs a modified subgraph $G'_{i+1}$ of $G_{i+1}$, which reduces the problem of finding a rank-maximal matching in $G_{i+1}$ to the problem of computing a maximum matching of $G'_{i+1}$. The modified subgraph $G'_{i+1}$ is constructed from $G'_i$ by adding only

**Algorithm 14** Algorithm Rank-Max

1: Start with $G'_1 = G_1$ and let $M_1$ be a maximum matching in $G'_1$.
2: **for** $i = 1$ to $z - 1$ **do**
3: Obtain an EOU labelling of $G'_i$.
4: Delete all edges in $G'_i$ connecting two odd vertices, or connecting an odd vertex with an unreachable vertex (this step does not delete any edge of $M_i$).
5: Delete from $E_j$, for all $j > i$, all edges incident to an odd or unreachable vertex in $G'_i$.
6: Build subgraph $G'_{i+1}$ by adding the edges in $E_{i+1}$ to $G'_i$.
7: Find a maximum matching $M_{i+1}$ in $G'_{i+1}$ by augmenting $M_i$.
8: **return** $M_z$ as a rank-maximal matching.

those edges from $E_{i+1}$ that can potentially belong to a rank-maximal matching of $I$.

To help us identify those edges that can potentially belong to a rank-maximal matching of $I$, we make use of the Edmonds-Gallai Decomposition as extended to the capacitated bipartite graph in Chapter 4. Recall that Lemma 4.3.4 shows that fundamental properties of the Edmonds-Gallai Decomposition, as introduced by Lemma 1.2.1 in Section 1.2, also hold in the capacitated bipartite graph case. Hence, we can reuse Lemma 4.3.4 here for each subgraph $G_i$ of $G$, which is essentially a CHAT instance. A pseudocode description of the algorithm is given in Algorithm 14.

### 7.4.2 Proof of correctness

Recall that Algorithm Rank-Max constructs a maximum matching $M_i$ for each subgraph $G'_i$. In order to show the correctness of our algorithm, we require the following technical results.

**Lemma 7.4.1.** *Suppose that every rank-maximal matching of $G_i$ is a maximum matching of $G'_i$. Then every rank-maximal matching of $G_{i+1}$ is contained in $G'_{i+1}$.*

**Lemma 7.4.2.** *For every phase $i$ and $j$ where $j > i$, the number of edges of rank at most $i$ is the same in $M_i$ and $M_j$.*

Note that the proofs of the lemmas have been omitted since they may be established by straightforward extension of the corresponding results for HAT (i.e. Lemmas 2.2 and 2.3 of [29] respectively). What Lemmas 7.4.1 and 7.4.2 show are the following:

(i) suppose that the hypothesis of Lemma 7.4.1 is true. Then, any edge deleted during phase $i$ of the algorithm does not belong to any rank-maximal matching of $G_{i+1}$ so that every rank-maximal matching in $G_{i+1}$ has all of its edges in $G'_{i+1}$;

(ii) the algorithm maintains the same number of edges of rank $i$ in each matching $M_j$ for all $i < j$.

The above results lead us to the following correctness result for our algorithm, for which we again omit the proof since it is a straightforward extension of the corresponding result for HAT (i.e. Theorem 2.4 of [29]).

**Theorem 7.4.1.** *For every $1 \leq k \leq z$, the following statements hold:*

*(i) Every rank-maximal matching in $G_k$ is a maximum matching in $G'_k$;*

*(ii) Every maximum matching $M_k$ in $G'_k$ is a rank-maximal matching in $G_k$.*

### 7.4.3 Time complexity analysis

The following theorem gives us the time complexity of our algorithm.

**Theorem 7.4.2.** *Given a* CHAT *instance $I$, a rank-maximal matching can be computed in $O(\min(z^*\sqrt{C}, C + z^*)m)$ time, where $z^*$ is the maximal rank of an edge in an optimal solution and $C$ is the total capacity of the houses in $I$.*

*Proof.* Consider phase $i$ of the algorithm. It is straightforward to see that each subgraph $G_i$ can be constructed in $O(m)$ time. We use Gabow's algorithm [15] to compute a maximum matching $M_i$ in $G_i$ in $O(\sqrt{C}m)$ time. Now, Gabow's algorithm uses successive augmentation steps to find a maximum matching for each $G_i$. It must be the case that the number of augmentation steps cannot exceed $|M_i| - |M_{i-1}| + 1$, since each step either increases the cardinality of the matching by at least 1, or establishes that no further steps are needed. Hence, the time complexity for Gabow's algorithm is also bounded by $O((|M_i| - |M_{i-1}| + 1)m)$, giving an overall bound of $O(\min(\sqrt{C}, |M_i| - |M_{i-1}| + 1)m)$.

After finding the maximum matching $M_i$, it follows that we can obtain an EOU labelling by using a similar approach to that described in Section 4.3.2 for CHAT. We first use a pre-processing step to label each unassigned agent and each undersubscribed house as even. Clearly, this step takes $O(n)$ time. Next, restricted breadth-first search may be used on $G_i$ to search for alternating paths with respect to $M_i$, building up odd or even labels for every vertex encountered. This step labels all odd and even (assigned) agents, and all odd and even (full) houses and takes $O(m)$ time. Any remaining unlabelled vertices must be unreachable and we can directly label these vertices in $G_i$ in $O(n)$ time. Thus,

the total time complexity of EOU labelling of $G_i$ is $O(n+m)$. The EOU labelling of $G_i$ is then used to delete certain edges from $E_j$ where $j > i$ which takes $O(m)$ time.

Hence, the time complexity for each phase is $O((\min(\sqrt{C}, |M_i| - |M_{i-1}| + 1)m)$. By summing, we see that the overall running time is $O(\min(z\sqrt{C}, C + z)m)$.

Now, we show how to replace $z$ by $z^*$. At the beginning of each phase $i$, we first check if $M_i$ is already a maximum matching in $G'$ which consists of all edges (of all ranks) that have not been deleted at the start of phase $i$. This takes $O(m)$ time. If $M_i$ is a maximum matching in $G'$, then we stop. Otherwise, we continue as described above. This ensures that only $z^*$ phases are executed. □

### 7.4.4 Alternative approaches to finding a rank-maximal matching

Given any CHAT instance $I$, we give here several alternative approaches for computing a rank-maximal matching in $I$. However, we will show that Algorithm Rank-Max, based on using the Edmonds-Gallai Decomposition, offers the fastest algorithm for all cases.

#### 7.4.4.1 Reduction to the Assignment problem

One method for finding a rank-maximal matching in $I$ would be by reduction to the Assignment problem through the allocation of a suitably steeply decreasing sequence of weights to the edges of $G$ as follows. For each (agent,house) edge $(a_i, h_j)$ in the underlying graph of $I$, let $wt(a_i, h_j) = (n+1)^{z-k}$ where $n$ is the total number of agents and houses, $z$ is the maximum length of an agent's preference list, and $rank_{a_i}(h_j) = k$. We then find a maximum weight matching in this weighted graph $G'$. The following lemma shows that such a maximum weight matching must be a rank-maximal matching in $I$.

**Lemma 7.4.3.** *Let $M$ be a maximum weight matching in $G'$. Then, $M$ is a rank-maximal matching of $I$.*

*Proof.* Suppose not. Then, there exists some matching $M'$ such that $M' \succ_L M$. Let $\rho_M = (x_1, ..., , x_z)$ and $\rho_{M'} = (y_1, ..., y_z)$. It follows that for some $s(1 \leq s \leq z)$, $x_i = y_i$ for each $i$ $(1 \leq i < s)$ and $y_s > x_s$. Now, let $wt(M^*)$ denote the weight of any matching $M^*$. Then, it follows that the weight of $M$ is

$$\begin{aligned} wt(M) &= \sum_{i=1}^{z} x_i(n+1)^{z-i} \\ &= \sum_{i=1}^{s} x_i(n+1)^{z-i} + \sum_{i=s+1}^{z} x_i(n+1)^{z-i} \\ &\leq \sum_{i=1}^{s} x_i(n+1)^{z-i} + (x_{s+1} + x_{s+2} + ... + x_z)(n+1)^{z-s-1} \\ &< \sum_{i=1}^{s} x_i(n+1)^{z-i} + n \cdot (n+1)^{z-s-1} \\ &< \sum_{i=1}^{s} x_i(n+1)^{z-i} + (n+1)^{z-s} \end{aligned}$$

However, we may derive the following inequality relation for the weight of $M'$:

$$\begin{aligned} wt(M') &\geq \sum_{i=1}^{s-1} y_i(n+1)^{z-i} + y_s(n+1)^{z-s} \\ &\geq \sum_{i=1}^{s-1} x_i(n+1)^{z-i} + (x_s+1)(n+1)^{z-s} \\ &\quad \text{(since } y_s \geq x_s + 1\text{)} \\ &= \sum_{i=1}^{s} x_i(n+1)^{z-i} + (n+1)^{z-s} \\ &> wt(M) \end{aligned}$$

which contradicts the fact that $M$ is a maximum weight matching in $G'$. □

Hence, Lemma 7.4.3 shows that we can find a rank-maximal matching in $I$ by reduction to the Assignment problem in the manner described above. Recall (from Section 1.2) that Gabow's algorithm for maximum weight DCS [15] solves the Assignment problem in the capacitated graph in $O(C \min(m \log n, n^2))$ time. In view of the time required to perform arithmetic operations on steeply decreasing weights [42], the resulting running time of the algorithm is $O(zC \min(m \log n, n^2))$. Recall also that Algorithm Rank-Max takes $O(\min(z^*\sqrt{C}, z^* + C)m)$ time. Now, if $m \log n \leq n^2$, then finding a rank-maximal matching via reduction to the Assignment problem takes (i) $\Omega(zCm \log n)$ time. Otherwise, this takes (ii) $\Omega(zCn^2)$ time. In case (i), Algorithm Rank-Max is faster by a factor of $\Omega(\min(\sqrt{C} \log n, (zC \log n)/(z^* + C)))$. In case (ii), Algorithm Rank-Max is faster by a factor of $\Omega(\min(\sqrt{C}, (zC)/(z^* + C)))$ under the standard assumption that $m \leq n^2$ [46]. It

follows in all cases that Algorithm Rank-Max is faster than by reduction to the Assignment problem.

We remark that a similar reduction to the Assignment problem can also be used to find greedy maximum matchings. Here, it is required to adjust the edge weights to ensure that a maximum weight matching in the weighted graph $G'$ described above also has maximum cardinality. In order to do so, we modify the previous reduction by adding a sufficiently large constant $C$ to each edge weight. This is to ensure that every matching of size $s$ has weight greater than every matching of size $s-1$. Since the largest edge weight we might have is $(n+1)^{z-1}$, adding a constant of $C=(n+1)^z$ is sufficient. Hence, we assign the weight of each edge $(a_i, h_j)$ to be $wt(a_i, h_j) = (n+1)^{z-k} + (n+1)^z$. Let $M$ be a maximum weight matching in the revised weighted graph $G'$. Then, the second component of the edge weights (as created by adding $C$ to the weight of every edge) ensures that $M$ will have maximum cardinality. Among all such matchings, the first component will ensure that a matching with maximum weight is a greedy maximum matching using a similar proof to Lemma 7.4.3.

It follows that such an approach would also take $O(zC\min(m\log n, n^2))$ time. Recall that Algorithm Greedy-Max takes $O(C^2mz)$ time to find a greedy maximum matching given a CHAT instance. For most practical applications, it is reasonable to assume that $C/\log n \geq 1$. For example, from the 2006-07 Scottish Foundation Allocation Scheme (SFAS) data [38], there were 781 students, 53 hospitals and the total capacity of the hospitals was 789, i.e. $n = 781 + 53 = 834$ and $C = 789$ so that $C/\log n \approx 117 \geq 1$. By comparing the time complexities of both approaches, it is straightforward to see that the reduction method is faster by a factor of $\Omega(C\max(1/\log n, m/n^2))$ for most practical cases.

#### 7.4.4.2 Adapting Algorithm Greedy-Max

Note that a rank-maximal matching and a greedy maximum matching are conceptually similar except that the former need not be of maximum cardinality. Now, a rank-maximal matching $M'$ must be a greedy $|M'|$-matching. The next lemma shows how to extend the results of Lemmas 7.3.1 and 7.3.2 to yield an alternative method for finding a rank-maximal matching in any instance of CHAT.

**Lemma 7.4.4.** *For each $i$, let $M_i$ be a greedy $i$-matching with profile $\rho_i$. If $\rho_{i+1} \succ_L \rho_i$ for $i = 1, 2, ..., k-1$ and $\rho_k \succ_L \rho_{k+1}$, then $M_k$ is a rank-maximal matching, and $\rho_k$ is the*

**Algorithm 15** Algorithm Greedy-Rank-Max

```
1: M := ∅;
2: M' := ∅;
3: s := 0;
4: loop
5:    P := a maximum profile augmenting path for M;
6:    if P exists then
7:       M := M ⊕ P;
8:       if ρ(M) ≻_L ρ(M') then
9:          M' := M;
10:      else {//ρ(M) ≺_L ρ(M') must hold}
11:         break;
12:   else
13:      break;
14:   s := s + 1;
15: M' is a rank-maximal matching;
```

*rank-maximal profile.*

*Proof.* Let $\rho_k = (x_1, ..., x_z)$ and suppose that $M_k$ is not rank-maximal. Then, there is a matching $M$ such that $M \succ_L M_k$ and $M$ has profile $\rho = (x_1, ..., x_{i-1}, y_i, ..., y_z)$ for some $i$ wih $y_i > x_i$. It follows that $M$ contains a matching $M'$ with profile $\rho' = (x_1, ..., x_{i-1}, x_i{+}1)$. Clearly, $M' \succ_L M_k$. Since $\rho_k = (x_1, ..., x_z)$, it follows that $x_1 + ... + x_z = k$, so that $x_1 + ... + x_i \leq k$. We then have that $|M'| = x_1 + ... + x_i + 1 \leq k + 1$. This implies that we have a matching of cardinality at most $k + 1$ whose profile left-dominates $\rho(M_k)$. However, this gives a contradiction since $\rho_k$ must left-dominate the profiles of all matchings of cardinality up to $k + 1$ by the statement of the lemma. □

We now introduce Algorithm Greedy-Rank-Max which offers an alternative approach to finding a rank-maximal matching in any given CHAT instance $J$. A pseudocode description of the algorithm is given in Algorithm 15. Algorithm Greedy-Rank-Max basically repeats the approach used by Algorithm Greedy-Max. However, the algorithm halts when it obtains a greedy $s$-matching that is also rank-maximal. That is, it does so when it finds a greedy $s$-matching $M$ which satisfies the condition in the statement of Lemma 7.4.4. It is straightforward to verify that Algorithm Greedy-Rank-Max has the same time complexity as Algorithm Greedy-Max, i.e. $O(C^2mz)$.

Lemma 7.4.4 shows that we can obtain a rank-maximal matching as a by-product

of finding a greedy maximum matching in $I$, and Algorithm Greedy-Rank-Max gives an algorithm for doing this. Let us compare the time complexities of Algorithm Rank-Max and Algorithm Greedy-Rank-Max. Recall Algorithm Rank-Max takes $O(\min(z^*\sqrt{C}, C + z^*)m)$ time. It follows that Algorithm Rank-Max is faster than Algorithm Greedy-Rank-Max by a factor of $\Omega(\min(C^{3/2}, C^2 z/(C + z^*))$.

#### 7.4.4.3 "Cloning"

Another straightforward solution to finding a rank-maximal matching for a CHAT instance $I$ may be to use "cloning" to create an instance $J$ of HAT, and then to apply the $O(\min(z^*\sqrt{n}, n + z^*)m)$ algorithm of [29] to $J$. Firstly, we create $c_j$ clones $h_j^1, h_j^2, ..., h_j^{c_j}$ of each house $h_j$ in $I$, where each clone has a capacity of 1. In addition, we replace each occurrence of $h_j$ in a given agent's preference list by the sequence $h_j^1, h_j^2, ..., h_j^{c_j}$, the elements of which are listed in a single tie at the point where $h_j$ appears.

Let us now compare the complexity of our direct approach using Algorithm Rank-Max to that of the cloning approach. Let $G_J$ denote the underlying graph of $J$. Then, $G_J$ contains $n' = n_1 + C$ vertices. For each $a_i \in A$, let $A_i$ denote the set of acceptable houses for $a_i$. Then, the number of edges in $G_J$ is $m' = \sum_{a_i \in A} \sum_{h_j \in A_i} c_j$. Hence, the complexity of applying the algorithm of [29] to $J$ is $\Omega(\min(z^*\sqrt{n_1 + C}, n_1 + C + z^*)m')$. It follows that the cloning approach is slower by a factor of $\Omega(m'/m)$.

## 7.5 Generous Maximum Matchings

For a given CHAT instance $I$, we say that a feasible $s$-profile $X$ is *$s$-right-minimal* if there is no other feasible $s$-profile that right-dominates $X$. In addition, we define a matching $M$ whose profile is $s$-right-minimal to be a *generous $s$-matching*. When $s$ is the cardinality of a maximum matching, we say that a generous $s$-matching is a *generous maximum* matching. As in the case of greedy $s$-matchings, there may be more than one generous $s$-matching for a given value of $s$, but it is clear that all generous $s$-matchings have the same profile, and we call this the *generous $s$-profile* for the problem instance. When $s$ is the cardinality of a maximum matching, the generous $s$-profile is called the *generous maximum profile*. We formalise the definition of a generous maximum matching as follows.

**Definition 7.5.1.** *Given an instance of* CHAT*, a generous maximum matching is a maximum matching that has minimum profile under the order $\prec_R$ taken over all matchings*

*in* $\mathcal{M}^+$.

### 7.5.1 Finding a generous maximum matching

We may adapt Algorithm Greedy-Max, designed to find a greedy maximum matching in $I$, into an analogous algorithm for finding a generous maximum matching in $I$. The remainder of this section will work towards showing this. First of all, we introduce the following lemma, which is a counterpart to Lemma 7.3.1 for finding generous matchings.

**Lemma 7.5.1.** *Let $M$ be a generous $s$-matching in $I$. Then either $M$ is a generous maximum matching or there is a generous $(s+1)$-matching $M'$ that can be obtained from $M$ via an augmenting path.*

*Proof.* It is straightforward to verify that if we replace all concepts relating to left-domination and greedy in the proof of Lemma 7.3.1 by their counterparts for right-domination and generous respectively, then this establishes the proof for this lemma. □

As with finding a greedy maximum matching in CHAT, given any generous $s$-matching $M$, we want to be able to identify an augmenting path with respect to $M$ that will lead us to a generous $(s+1)$-matching. To do so, let us introduce the notion of a *minimum profile augmenting path*. Let $i$ be an integer such that $1 \leq i \leq z$. We assume $X + i$ and $X - i$ to be those operations on $z$-tuples and $i$ that were defined previously in the context of greedy maximum matchings. Then, for every house vertex $h_j$, we define the *R-value* of $h_j$ with respect to $M$, denoted by $R(h_j)$, to be the minimum profile taken over all alternating paths from an exposed agent vertex ending at $h_j$. We say that an alternating path $P$ is a *minimum profile* augmenting path for $M$ if $P$ is an augmenting path, and $\rho(P) = \min\{R(h_j) : h_j \in H\}$ where min is with respect to the $\prec_R$ order on profiles.

The following lemma, analogous to Lemma 7.3.2, shows that we can use the notion of a minimum profile augmenting path in tandem with the classical augmenting path algorithm to find a generous maximum matching in CHAT.

**Lemma 7.5.2.** *Suppose that $M$ is a generous $s$-matching which is not maximum. Let $P$ be a minimum profile augmenting path. Then, augmenting $M$ along $P$ gives a generous $(s+1)$-matching.*

*Proof.* It is straightforward to verify that if we replace all concepts relating to left-domination and greedy in the proof of Lemma 7.3.2 by their counterparts for right-domination and generous respectively, then this establishes the proof for this lemma. □

Hence, we are able to reuse Algorithm Greedy-Max to find a generous maximum matching for a given CHAT instance $I$. The difference, in this context, is that we are interested in finding a minimum profile augmenting path in each iteration of the algorithm instead. Let $M$ be any generous $s$-matching that is not a generous maximum matching for a given CHAT instance $I$. If we replace all occurrences of $L$-values in Algorithm Max-Aug by $R$-values, and replace the left-domination comparison $\succ_L$ by the right-domination comparison $\prec_R$ in line 6 of the algorithm, then we can reuse Algorithm Max-Aug to find a minimum profile augmenting path with respect to $M$. Note that if there does not exist any alternating path of length 1 from an exposed agent vertex to a house $h_j$ in the initialisation step, then we set $r(h_j)$ to be $O'$ where $r(h_j)$ is used to compute $R(h_j)$ in the algorithm. Let us rename Algorithm Max-Aug, after the above transformations, to be Algorithm Min-Aug. Then, if every house vertex has an $R$-value that is equal to $O'$ after execution of Algorithm Min-Aug, then there is no augmenting path, and $M$ is a generous maximum matching. Otherwise, we find the house vertex $h_j$ with minimum $R$-value, and obtain the minimum profile augmenting path $P$ by alternately tracing the predecessor values and matched edges in $M$ starting from $pred(h_j)$.

### 7.5.2 Proof of correctness

As in the greedy maximum case, we want to show that the augmenting path $P$ obtained by executing Algorithm Min-Aug is a minimum profile augmenting path with respect to the current generous $s$-matching $M$. The next two lemmas prove results analogous to the greedy maximum case in the generous maximum context.

**Lemma 7.5.3.** *When Algorithm Min-Aug terminates, $r(h_j) = R(h_j)$ for each house vertex $h_j \in H$.*

*Proof.* We replace all concepts relating to left-domination and greedy by their counterparts for right-domination and generous respectively in the proof of Lemma 7.3.3 to obtain our proof for this lemma. □

**Lemma 7.5.4.** *Let $P$ be the sequence of agents and houses obtained by alternately tracing predecessor values and matched edges in $M$ starting from $pred(h_j)$ where the $R$-value of $h_j$ right-dominates the $R$-values of all exposed house vertices. Then, $P$ terminates at some exposed agent vertex.*

*Proof.* We again replace all concepts relating to left-domination and greedy by their counterparts for right-domination and generous respectively in the proof of Lemma 7.3.4 to obtain our proof for this lemma. □

This gives us the following result with respect to generous maximum matchings in CHAT.

**Theorem 7.5.1.** *Let $M$ be a generous s-matching $M$ that is not generous maximum. Then, executing Algorithm Min-Aug (adapted from Algorithm Max-Aug as described above) finds a minimum profile augmenting path with respect to $M$.*

### 7.5.3 Time complexity analysis

It is straightforward to verify that it takes $O(C^2mz)$ time to find a generous maximum matching using the same arguments as in the greedy maximum case. In practice, as in the generous maximum case, we can speed up the running time by halting Algorithm Min-Aug when no house $h$ that is assigned in the current matching had $r(h)$ updated in the last iteration of the algorithm.

Finally, we remark that we can find a generous maximum matching given an instance $I$ of CHAT by a reduction to the Assignment problem in a similar way to that for the greedy maximum case. In the generous maximum case, however, the appropriate weight to assign to each edge $(a_i, h_j)$ should be $wt(a_i, h_j) = ((n+1)^{z-1} - (n+1)^{k-1} + 1) + (n+1)^z$ where $(n+1)^z$ is again the large constant added to ensure that any maximum weight matching in the underlying weighted graph has maximum cardinality. As with the greedy maximum case, such an approach would take $O(zC \min(m \log n, n^2))$ time. Furthermore, we can assume that $C/\log n \geq 1$ for most practical applications. Hence, the reduction method is again faster than an augmenting path approach using the Bellman-Ford algorithm by a factor of $\Omega(C \max(1/\log n, m/n^2))$ for most practical cases.

## 7.6 Open Problems

We conclude this chapter with the following open problems.

- In this chapter, we have given different efficient algorithms for the individual problems of finding a greedy maximum, a rank-maximal and a generous maximum matching given an CHAT instance. It was observed in Section 7.4 that an approach util-

ising the Edmonds-Gallai Decomposition gives us a faster algorithm for finding a rank-maximal matching than several straightforward alternatives such as by an augmenting path approach utilising the Bellman-Ford algorithm or by reduction to the Assignment problem. We have also shown that the reduction approach gives a faster algorithm than the augmenting path approach utilising the Bellman-Ford algorithm for constructing greedy and generous maximum matchings for most practical cases. Hence, the question arises as to whether we can find direct, more efficient algorithms for constructing greedy and generous maximum matchings, given an instance of CHAT, by also making use of the Edmonds-Gallai Decomposition.

- For a given instance $I$ of CHAT, and a given profile, can we determine whether $I$ admits a matching with that profile in polynomial time?

# Chapter 8

# Profile-based optimal matchings in HRT

## 8.1 Introduction

In Chapter 7, we studied three different types of profile-based optimal matchings in the context of CHAT. These are the concepts of a greedy maximum matching, a rank-maximal matching and a generous maximum matching. In this chapter, we extend these results to the bipartite matching problem model with two-sided preference lists, focusing on the Hospital-Residents problems with Ties (HRT). The definitions of a greedy maximum, rank-maximal and generous maximum matching respectively are the same as those given in Chapter 7. Furthermore, we reuse most of the terminology and notation as defined in Chapter 7, and we explicitly define here the relevant concepts only where we need to adapt them to HRT.

The main results of this chapter, and their organisation are as follows. First of all, Section 8.2 introduces the terminology and notations that will be used for the rest of this chapter. We then show how to find a greedy maximum matching, a rank-maximal matching and a generous maximum matching given an instance of HRT in Sections 8.3, 8.4 and 8.5 respectively. In each of these sections, we give two algorithms, one based on the augmenting path approach utilising the Bellman-Ford algorithm, and the other through reduction to the Assignment problem (both as described in Chapter 7), to give efficient solutions to the problem. Finally, since SMTI is a special case of HRT, we remark that the algorithms described in this chapter can also be used to solve the analogous problems in SMTI.

## 8.2 Basic terminology

Let $I$ be an instance of the *Hospitals-Residents problem with Ties* (HRT). This comprises two disjoint sets $R$ and $H$, 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. We assume all the terminology and definitions that were introduced for HR in Chapter 1. Moreover, we now allow the preference list of each agent to contain ties. Let $G = (R, H, E)$ be the underlying graph of $I$ where $E$ is the set of edges in $G$ representing the acceptable hospitals of the residents (and vice versa). Recall that $C = \sum_{j=1}^{n_2} c_j$ denotes the sum of the capacities of the hospitals. We assume that no resident or hospital has an empty preference list so that $m = |E| \geq \max\{n_1, n_2\}$.

Given a resident $r_i \in R$ and an acceptable partner $h_j$ for $r_i$, we define $rank_{r_i}(h_j)$ to be the number of hospitals that $r_i$ prefers to $h_j$ plus 1. If $rank_{r_i}(h_j) = k$, we say that $h_j$ is a *kth choice* of $r_i$. In a similar way, we define $rank_{h_j}(r_i)$ and a $k$th choice of $h_j$. Let $z$ be the largest rank of a resident or hospital taken over all preference lists in $I$. Let $(r_i, h_j) \in E$ be any edge. Then, we define the rank of $(r_i, h_j)$ to be the pair $r(r_i, h_j) = (\min\left\{rank_{h_j}(r_i), rank_{r_i}(h_j)\right\}, \max\left\{rank_{h_j}(r_i), rank_{r_i}(h_j)\right\})$.

Given a matching $M$ for an instance $I$ of HRT, we define a vertex $v$ in the underlying graph $G$ of $I$ to be *exposed* with respect to $M$, if $v$ is a resident vertex that is unassigned in $M$, or if $v$ is a hospital vertex that is undersubscribed in $M$. An *augmenting path* in $G$ is an alternating path both of whose end vertices are exposed.

For a given HRT instance $I$, the definition of a *feasible s-profile* $X = (x_1, ..., x_z)$ is analogous to that in CHAT. However, in HRT, $\sum x_i = 2s$. Let $\mathcal{M}$ be the set of all matchings in $I$. Furthermore, let $\mathcal{M}^+$ denote the set of maximum matchings in $\mathcal{M}$.

## 8.3 Greedy Maximum matchings

Let $\succ_L$ be the total order defined on profiles of matchings as in Section 7.2. The following formalises the definition of a greedy maximum matching with respect to HRT.

**Definition 8.3.1.** *Given an instance of* HRT*, a greedy maximum matching is a maximum matching that has maximum profile under the order $\succ_L$ taken over all matchings in $\mathcal{M}^+$.*

### 8.3.1 Finding a greedy maximum matching

Given any instance $I$ of HRT, we may find a greedy maximum matching in $I$ by using Algorithm Greedy-Max, which was used to solve the analogous problem in CHAT. We

first introduce the following lemma, which is analogous to Lemma 7.3.1 for finding greedy matchings in the CHAT case.

**Lemma 8.3.1.** *Let $M$ be a greedy $s$-matching in $I$. Then either $M$ is a greedy maximum matching or there is a greedy $(s+1)$-matching $M'$ that can be obtained from $M$ via an augmenting path.*

*Proof.* The proof of Lemma 7.3.1 may be adapted to prove this lemma. □

As with finding a greedy maximum matching in CHAT, given any greedy $s$-matching $M$ in $I$, we want to be able to identify an augmenting path with respect to $M$ that will lead us to a greedy $(s+1)$-matching. To do so, we extend the notion of a *maximum profile augmenting path* from CHAT to HRT. Recall the concept of left-domination on $z$-tuples as defined in Chapter 7. That is, given the $z$-tuples $X = (x_1, ..., x_z)$ and $Y = (y_1, ..., y_z)$, we say that $Y \succ_L X$, or $Y$ *left-dominates* $X$, if there exists some $k$ $(1 \leq k \leq z)$ such that $x_i = y_i$ for $1 \leq i < k$ and $y_k > x_k$. Let $(a, b)$ be a pair of integers such that $1 \leq a \leq b \leq z$. We then define the following operations on $z$-tuples and $(a, b)$. That is, if $a < b$, we then define $X + (a, b)$ to be

$$X + (a, b) = (x_1, ..., x_{a-1}, x_a + 1, x_{a+1}, ..., x_{b-1}, x_b + 1, x_{b+1}, ..., x_z)$$

and we define $X - (a, b)$ to be

$$X - (a, b) = (x_1, ..., x_{a-1}, x_a - 1, x_{a+1}, ..., x_{b-1}, x_b - 1, x_{b+1}, ..., x_z)$$

Otherwise, $a = b$, and we define $X + (a, b)$ to be

$$X + (a, b) = (x_1, ..., x_{a-1}, x_a + 2, x_{a+1}, ..., x_z)$$

and we define $X - (a, b)$ to be

$$X - (a, b) = (x_1, ..., x_{a-1}, x_a - 2, x_{a+1}, ..., x_z)$$

Let $P = \langle r_0, h_0, r_1, h_1, ..., r_x, h_x \rangle$ be an alternating path from an exposed resident vertex $r_0$ to a hospital vertex $h_x$, such that $(r_i, h_{i-1}) \in M$ for $1 \leq i \leq x$. We then define the *profile* of $P$ to be

$$\begin{aligned} \rho(P) \quad = \quad & O + r(r_0, h_0) + r(r_1, h_1) + ... + r(r_x, h_x) \\ & - r(r_1, h_0) - r(r_2, h_1) - ... - r(r_x, h_{x-1}) \end{aligned}$$

where $O = (o_1, ..., o_z)$ is the $z$-tuple such that $o_i = 0$ for $1 \leq i \leq z$. It follows that if $P$ is an augmenting path, then $\rho(P)$ corresponds to the net change in the profile of $M$ if we augment $M$ along $P$. For every hospital vertex $h_j$, we define the *L-value* of $h_j$ relative to $M$, denoted by $L(h_j)$, to be the maximum profile taken over all alternating paths from an exposed resident vertex ending at $h_j$. We say that an alternating path $P$ is a *maximum profile* augmenting path for $M$ if $P$ is an augmenting path, and $\rho(P) = \max\{L(h_j) : h_j \in H\}$ where max is with respect to the $\succ_L$ order on profiles.

Let $M$ be any greedy $s$-matching that is not a greedy maximum matching for a given HRT instance $I$. The following lemma, analogous to Lemma 7.3.2, shows us that we can use a maximum profile augmenting path to obtain a greedy $(s+1)$-matching from $M$.

**Lemma 8.3.2.** *Suppose that $M$ is a greedy $s$-matching which is not maximum. Let $P$ be a maximum profile augmenting path. Then, augmenting $M$ along $P$ gives a greedy $(s+1)$-matching.*

*Proof.* The proof of Lemma 7.3.2 may be adapted here. □

It now remains to show how to find a maximum profile augmenting path with respect to $M$. We remark that we may reuse Algorithm Max-Aug from Chapter 7 for this by making the following changes to the algorithm:

- replace the agent vertices and house vertices in Algorithm Max-Aug by resident and hospital vertices respectively; and
- replace the arithmetic operations used by the edge relaxtion operation in line 5 of Algorithm Max-Aug by those defined with respect to $z$-tuples and integer pairs in this chapter, instead of those defined in Chapter 7.

As in CHAT, if every hospital vertex has a $L$-value that is (0) after execution of Algorithm Max-Aug, then there is no augmenting path, and $M$ is a greedy maximum matching.

### 8.3.2 Proof of correctness

Let $I$ be an instance of HRT. Let $P$ be the sequence of residents and hospitals obtained by alternately tracing predecessor values and matched edges in $M$ starting from $pred(h_r)$ where the hospital vertex $h_r$ has left-maximum $L$-value over all exposed hospital vertices

after an execution of Algorithm Max-Aug (as modified above). As in the CHAT case, we want to show that the augmenting path $P$ is a maximum profile augmenting path with respect to the current greedy $s$-matching $M$. The next two lemmas prove the analogous results for the CHAT case in the context of HRT.

**Lemma 8.3.3.** *Let Algorithm Max-Aug (as modified above) be executed. When Algorithm Max-Aug terminates, $l(h_j) = L(h_j)$ for each hospital vertex $h_j \in H$.*

*Proof.* The proof of Lemma 7.3.3 may be adapted here. □

**Lemma 8.3.4.** *Let $P$ be the sequence of residents and hospitals obtained by alternately tracing predecessor values and matched edges in $M$ starting from $pred(h_r)$ where the hospital vertex $h_r$ has left-maximum $L$-value over all exposed hospital vertices. Then, $P$ terminates at some exposed resident vertex.*

*Proof.* The proof of Lemma 7.3.4 may be adapted here. □

Lemmas 8.3.3 and 8.3.4 give us the following result.

**Theorem 8.3.1.** *Let $M$ be a greedy $s$-matching $M$ that is not greedy maximum. Let Algorithm Max-Aug (as modified above) be executed. Then, the algorithm finds a maximum profile augmenting path with respect to $M$.*

### 8.3.3 Time complexity analysis

The time complexity for finding a greedy maximum matching in an instance $I$ of HRT may be easily verified to be the same as that for solving the same problem in any instance of CHAT, that is $O(C^2mz)$, where $C$ is the total capacity of the hospitals, $m$ is the total length of preference lists and $z$ is the maximum length of any preference list respectively in $I$. As with CHAT, we can speed up the actual runtime of Algorithm Greedy-Max through the observation that if no hospital had its $L$-value updated in the last iteration of Algorithm Max-Aug, then no further improvement to an $L$-value can happen. Hence, we may choose to halt Algorithm Max-Aug at that point.

Recall that a straightforward approach to constructing a greedy maximum matching in the setting of CHAT was by reduction to the Assignment problem. We remark that it is possible to reuse a similar reduction to that described in Section 7.4.4.1 to also find a greedy maximum matching in the context of HRT as follows. Let $(r_i, h_j)$ be an edge where $rank_{r_i}(h_j) = k$ and $rank_{h_j}(r_i) = l$. Then, we let $wt(r_i, h_j) = (n+1)^{z-k} + (n+1)^{z-l}$.

We are then interested in finding a maximum weight matching $M$ in this weighted graph $G'$ that has maximum cardinality. To ensure that $M$ has maximum cardinality, we add a sufficiently large constant to the edge weights as in the context of CHAT. Here, the largest edge weight that we might have is $2(n+1)^{z-1}$, so adding a constant of $2(n+1)^z$ is sufficient. Then, it is straightforward to reuse the arguments for the analogous case in CHAT to verify that a maximum weight matching $M$ in $G'$ must be a maximum cardinality matching, and is a greedy maximum matching for $I$. Since the underlying graph of $I$ is capacitated, we require to use Gabow's algorithm for maximum weight DCS again to find a maximum weight matching in $G'$. Hence, we can find a greedy maximum matching in $I$ in $O(zC\min(m\log n, n^2))$ time by reduction to the Assignment problem. Comparing this to the approach using Algorithm Greedy-Max, it is straightforward to see that the reduction approach is again faster by a factor of $\Omega(C\max(1/\log n, m/n^2))$ for most practical cases.

## 8.4 Rank-maximal matchings

Let $I$ be an instance of HRT. We formalise the definition of a rank-maximal matching in $I$ as follows.

**Definition 8.4.1.** *Given an instance of* HRT*, a rank-maximal matching is a matching that has maximum profile under the order $\succ_L$ taken over all matchings in $\mathcal{M}$.*

### 8.4.1 Finding a rank-maximal matching

As in the case of greedy maximum matchings, we can find a rank-maximal matching in a given HRT instance $I$ either by constructing an algorithm based on the augmenting path approach utilising the Bellman-Ford algorithm or by reduction to the Assignment problem. In the former case, the next lemma shows that we can use this approach by extending the results of Lemmas 8.3.1 and 8.3.2 in a subtle way to obtain the analogue of Lemma 7.4.4.

**Lemma 8.4.1.** *For each $i$, let $M_i$ be a greedy $i$-matching with profile $\rho_i$. If $\rho_{i+1} \succ_L \rho_i$ for $i = 1, 2, ..., k-1$ and $\rho_k \succ_L \rho_{k+1}$, then $M_k$ is a rank-maximal matching, and $\rho_k$ is the rank-maximal profile.*

*Proof.* Let $\rho_k = (x_1, ..., x_z)$ and suppose that $M_k$ is not rank-maximal. Then, there is a matching $M$ such that $M \succ_L M_k$. Let $\rho = (x_1, ..., x_{i-1}, y_i, ..., y_z)$ be the profile of $M$. It

follows that $y_i > x_i$ for some $i$. Now, it must be the case that $k+2 \leq |M| \leq C$ since $M \succ_L M_k$ but $\rho_k$ left-dominates the profiles of all matchings of cardinality 1 to $k+1$. Let $G$ be the underlying bipartite graph of $I$. Let us consider $X = M_k \oplus M$. Then, it follows that each connected component of $X$ is either (i) an odd length alternating path, (ii) an even length alternating path or (iii) an alternating cycle. Since $M \succ_L M_k$, there must exist at least one connected component $C$ of $X$ such that $\rho_C(M) \succ_L \rho_C(M_k)$.

Suppose that $C$ is of type (i). Let $M' = M_k \oplus C$. Clearly, $M' \succ_L M_k$. Now, the end edges of $C$ cannot be in $M_k$, for otherwise $M'$ is a $(k-1)$-matching which left-dominates $M_k$, a contradiction. Hence, the end edges of $C$ must be in $M$. It follows that $C$ is then an augmenting path with respect to $M_k$. However, this implies that $M'$ is a $(k+1)$-matching which left-dominates $M_k$, which is a contradiction by the statement of the lemma.

Hence, suppose that $C$ is of either type (ii) or (iii). Since $\rho_C(M) \succ_L \rho_C(M_k)$, we can create a new matching $M'$ of cardinality $k$ by replacing the $M_k$-edges in $C$ by the $M$-edges in $C$, giving $\rho(M') \succ_L \rho(M_k)$, which is a contradiction since $M_k$ is a greedy $k$-matching. □

What Lemma 8.4.1 thus implies is that we can reuse the approach of Algorithm Greedy-Rank-Max to find a rank-maximal matching in $O(C^2mz)$ time given an instance $I$ of HRT. As mentioned above, an alternative approach to find a rank-maximal matching in $I$ may be by reduction to the Assignment problem. To do so, we reuse the reduction as described in Section 8.3 for finding a greedy maximum matching by making just one change. Since we do not require to find a maximum weight matching in the underlying graph that must be of maximum cardinality, we remove the large constant that was added to the weight of each edge. Then, it is straightforward to verify that Lemma 7.4.3 also holds in the HRT case. Hence, any maximum weight matching $M$ in the weighted graph is also a rank-maximal matching for $I$. As with the greedy maximum case, it follows that we can find a rank-maximal matching in $I$ in $O(zC \min(m \log n, n^2))$ time by reduction to the Assignment problem.

By comparing the time complexities of the augmenting path approach based on the Bellman-Ford approach and the reduction to the Assignment problem, it is straightforward to see that the latter is faster than the former by a factor of $\Omega(C \max(1/\log n, m/n^2))$ for most practical cases.

## 8.5 Generous Maximum Matchings

Given an instance $I$ of HRT, an alternative optimality criteria as opposed to looking for greedy maximum matchings or rank-maximal matchings may be to look for generous maximum matchings. Let $\prec_R$ be the total order defined on profiles of matchings as in Section 7.2. We formalise the definition of a generous maximum matching as follows.

**Definition 8.5.1.** *Given an instance of* HRT*, a generous maximum matching is a maximum matching that has minimum profile under the order $\prec_R$ taken over all matchings in $\mathcal{M}^+$.*

### 8.5.1 Finding a generous maximum matching

Now, as with the greedy maximum and rank-maximal case, we have two possible algorithms for constructing a generous maximum matching based on the augmenting path approach utilising the Bellman-Ford algorithm and reduction to the Assignment problem.

In the former case, it may be verified that the results of Lemmas 7.5.1-7.5.4 and Theorem 7.5.1 can be extended from CHAT to HRT using the same proofs that established these results in the CHAT case. What this shows is that we can reuse the algorithm for finding a generous maximum matching in CHAT for the analogous problem in HRT. That is, we start from the empty matching, and then use Algorithm Greedy-Max, as transformed for finding a generous maximum matching in CHAT, to repeatedly increase the cardinality of the current generous matching in stages. In each stage, we use Algorithm Min-Aug, as transformed for finding a minimum profile augmenting path in CHAT, for finding a similar type of augmenting path to augment the current generous matching $M$ in $I$ until no such path can be found. When this happens, $M$ is a generous maximum matching for $I$. Since the algorithm for finding a generous maximum matching in HRT is then the same as that for solving the analogous problem in CHAT, the time complexity for this is also $O(C^2mz)$. As in the CHAT case, we can speed up the runtime by halting Algorithm Min-Aug when no hospital which is assigned in the current generous matching had its $R$-value updated in the last iteration of the algorithm.

To reduce the problem of finding a generous maximum matching to the Assignment problem, the correct edge weight[1] to use for each edge $(r_i, h_j)$ should be $wt(r_i, h_j) = (2(n+1)^{z-1} - (n+1)^{k-1} - (n+1)^{l-1} + 1) + 2(n+1)^z$. We can then easily extend the results

[1]The term +1 is to ensure that all edge weights are positive.

for generous maximum matchings in the CHAT case to show that a maximum weight matching in the weighted graph as constructed above also gives a generous maximum matching in $O(zC\min(m\log n, n^2))$ time in the context of HRT.

Finally, it is straightforward to verify that the reduction approach is again faster by a factor of $\Omega(C\max(1/\log n, m/n^2))$ for most practical cases.

## 8.6 Profile-based optimal matchings in SMTI

Since SMTI is a special case of HRT, we observe that each of the algorithms described above for finding a greedy maximum matching, a rank-maximal matching and a generous maximum matching can also be used to find a matching of the same kind given an SMTI instance $I$. We remark that in the case of the augmenting path approach based on the Bellman-Ford algorithm, since the maximum cardinality of a matching in $I$ is $O(n)$, we only need to replace the $C$ factor in the time complexity of the algorithm in order to obtain its respective running times in $I$, i.e. $O(n^2mz)$. In the case of reduction to the Assignment problem, it follows that we can use the fastest algorithm for finding a maximum weight matching in an uncapacitated bipartite graph since all agents in $I$ have capacity 1. Recall from Section 1.2 that this takes $O(nm + n^2\log n)$ time [14]. In view of the time required to perform arithmetic operations on steeply decreasing weights [42], the resulting running time of the algorithm is $Oz(nm + n^2\log n)$. It follows that if $nm \leq n^2\log n$, then the reduction gives a faster solution by a factor of $\Omega(m/\log n)$. Otherwise, the reduction is faster by a factor of $\Omega(n)$. Hence, as in the case of HRT, reduction to the Assignment problem gives a faster solution to the problem than an augmenting path approach based on the Bellman-Ford algorithm in SMTI.

## 8.7 Open Problems

We conclude this chapter with the following open problems.

- It was observed in Chapter 7 that an approach utilising the Edmonds-Gallai Decomposition led to a faster algorithm for finding a rank-maximal matching given a CHAT instance than alternatives such as using an augmenting path approach based on the Bellman-Ford algorithm or by reduction to the Assignment problem. This raises the question as to whether we can also obtain similar results for the case of HRT.

- Each of the open problems posed for CHAT at the end of Chapter 7 could be naturally extended to HRT. Can we then find solutions to these problems in the context of HRT?

# Chapter 9

# Conclusion

## 9.1 Summary of thesis contribution

The contribution of this thesis can be divided into three main classes based on three different types of optimality criteria that can be applied to a matching in the context of a bipartite matching problem with preferences. These are the concepts of

1. a Pareto optimal matching
2. a popular matching
3. a profile-based optimal matching, which may be sub-divided into the concepts of
   (a) a rank-maximal matching
   (b) a greedy maximum matching
   (c) a generous maximum matching

For each optimality criterion, we first studied the concept in the setting where preferences are only one-sided (i.e. the cases of CHA, WCHA or CHAT as appropriate), and then extended the results to allow preferences to be two-sided (i.e. the cases of HR, SMI, HRT or SMTI as appropriate). Figure 9.1 sumarises the main contributions of this thesis according to this classification by giving the fastest algorithm for computing an optimal matching in each case. In the table, the second column indicates the type of preference lists in the problem instance, by using a 's' to indicate strict preferences and a 't' to indicate that ties are allowed. In addition, we indicate the chapter number of this thesis in square brackets, where the results of each algorithm can be found. We remark

| Optimality Criterion | | Capacitated, one-sided preferences (i.e. CHA) | Weighted, capacitated, one-sided preferences (i.e. WCHA) | Uncapacitated, two-sided preferences (i.e. SMI) | Capacitated, two-sided preferences (i.e. HR) |
|---|---|---|---|---|---|
| Pareto Optimal | s | $O(\sqrt{C}m$ [2] | N/A | $O(\sqrt{n}m)$ [3] | $O(\sqrt{C}m)$ [3] |
| | t | $O(C\min(m\log n, n^2))$ [2] | N/A | $O(nm + n^2\log n)$ [3] | open |
| Popular | s | $O(\sqrt{C}n_1 + m)$ [4] | $O(\sqrt{C}n_1 + m)$ [5] | open | open |
| | t | $O(\sqrt{C}m)$ [4] | open | open | open |
| Rank-maximal | s/t | $O(\min(z^*\sqrt{C}, C + z^*)m)$ [7] | open | $Oz(nm + n^2\log n)$ [8] | $O(zC\min(m\log n, n^2))$ [8] |
| Greedy maximum | s/t | $O(zC\min(m\log n, n^2))$ [7] | open | $Oz(nm + n^2\log n)$ [8] | $O(zC\min(m\log n, n^2))$ [8] |
| Generous maximum | s/t | $O(zC\min(m\log n, n^2))$ [7] | open | $Oz(nm + n^2\log n)$ [8] | $O(zC\min(m\log n, n^2))$ [8] |

Figure 9.1: Complexity of algorithms for producing optimal matchings in bipartite matching problems with preferences.

**Key:**

$C$: the total capacity of the houses in a CHA or WCHA instance or the total capacity of the hospitals in a HR instance

$m$: the total length of the preference lists in the problem instance

$n$: the total number of participants in the problem instance

$n_1$: total number of agents in a CHA or WCHA instance

$z$: maximum length of an agent's preference list in the problem instance

$z^*$: maximal rank used in any optimal solution in the problem instance

that all the results mentioned in Figure 9.1 are new and contained in this thesis. With reference to this figure, we now present some conclusions regarding the results contained in this thesis.

## 9.2 Pareto optimal matchings

1. Figure 9.1 indicates that a maximum Pareto optimal matching can be found in polynomial time given an instance of CHA, SMI or HR. The time complexities of our algorithms are each bounded above by the time taken to find a maximum matching in the underlying graph of the problem instance. Hence, these algorithms may be considered efficient in the sense that any improvement to their time complexities would imply an improved algorithm for finding a maximum matching in a bipartite graph (capacitated or uncapacitated as appropriate).

2. As can be seen from Figure 9.1, the ties case in HR is still open. Given that a combinatorial approach gave a faster solution for finding a maximum Pareto optimal matching in each of the listed problems where preferences are strict, and given the close relationship of these problems to one another, it is likely that a combinatorial approach to one of the problems in the presence of ties would also yield faster solutions to each of these problems.

## 9.3 Popular matchings

1. The problem of finding a maximum popular matching, or determining that none exists, can be solved in polynomial time given an instance of CHAT or WCHA.

2. We remark that it is likely that a polynomial-time algorithm for finding a maximum popular matching, or determining that none exists, given an instance of WCHAT, can be obtained by extending in a natural way the algorithm for WCHA, in much the same way as our algorithm for CHAT was extended from that for CHA.

3. Finding an algorithm to construct a maximum popular matching in a given bipartite matching problem with two-sided preferences remains open. However, we can test whether any matching in a given SMTI-SYM instance is popular using the $O\sqrt{n}m)$ time algorithm described in Chapter 6. Furthermore, it is likely that any combina-

torial solution to the problem would be required to use our characterisation results for popular matchings in SMTI-SYM as presented in Chapter 6.

## 9.4 Profile-based optimal matchings

1. Each of a rank-maximal, a greedy maximum and a generous maximum matching can be found in polynomial time given an instance of CHAT, HRT or SMTI.

2. It is faster to find a rank-maximal matching given an instance of CHAT if the underlying approach makes use of the Edmonds-Gallai Decomposition rather than straightforward alternatives such as an augmenting path approach based on the Bellman-Ford algorithm or by reduction to the Assignment problem. Hence, it is likely that any combinatorial approach utilising the Edmonds-Gallai Decomposition would also offer a faster algorithm for each of the problems of finding a greedy maximum and a generous maximum matching in an instance of CHAT. Indeed, the viability of the augmenting path approach based on the Bellman-Ford algorithm suggests such a possibility since the approach using the Edmonds-Gallai Decomposition is also inherently based on augmenting paths.

3. The above observations and results are also likely to extend naturally to the problems of finding a rank-maximal, a greedy maximum and a generous maximum matching in an instance of HRT or SMTI.

## 9.5 General observations

We make some conclusions here on the results of this thesis in general, in addition to those specific to each of the optimality criterion studied as above.

1. All the problems that were studied in this thesis turned out to be solvable in polynomial time. The existence of a polynomial-time algorithm is often inherently associated with establishing some kind of underlying structure for the problem concerned. For instance, the solution to finding a maximum Pareto optimal matching requires the identifying and then satisfying of certain types of coalitions in the underlying graph of the problem instance. To find a maximum popular matching, the identification of the $f$- and $s$-partners of each participating agent in the problem instance allows us to restrict our attention to only a subgraph of the underlying graph to

generate an efficient solution to the problem. In addition, the fastest algorithm for finding a rank-maximal matching is reliant on utilising the Edmonds-Gallai Decomposition to label vertices and then identify only those edges that can belong to any solution.

2. The optimality criteria studied in this thesis can be considered to be superficially similar in some respects but a simple change to the problem definition often requires a significant change to the algorithm.

3. The problem of finding a matching of maximum cardinality in the underlying graph of the problem instance often seems to influence the time complexity of the resulting algorithm for its solution. This is the case even if the underlying problem seems on the surface not to be associated with maximum matchings, e.g. popular matchings.

## 9.6 Future work

There is a wide range of possibilities for future study beyond the problems considered in this thesis. These include the open problems listed at the end of each of the preceding chapters, as well as the following.

1. Can we establish any further structural results for the sets of Pareto optimal matchings for a given instance of CHA or HR? The same question arises for each of the other optimality criteria studied in this thesis.

2. For a given bipartite matching problem, there may be many different matchings of a certain type, e.g. Pareto optimal, popular, rank-maximal etc. However, it is open as to whether we can find algorithms to efficiently generate all matchings of a given kind. Towards this, we note that Uno [62] has given algorithms for generating all the perfect, maximum and maximal matchings in a bipartite graph, so that any efficient algorithm could possibly extend his approach. As a first step, given a bipartite matching problem and an optimality criterion to satisfy for a matching in the problem instance, it would be useful to find efficient algorithms to determine whether the problem instance admits a unique matching of the required type, and if not to find a second such matching.

3. From a practical point of view, perform empirical analyses of the algorithms presented in this thesis in order to gain a greater degree of understanding of their

behaviour in real-life situations, e.g. how "good" are these different kinds of matchings likely to be for a given instance of CHA, HR etc. derived from some practical application?

# Bibliography


[1] A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3)(3):689–701, 1998.

[2] A. Abdulkadiroğlu and T. Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88:233–260, 1999.

[3] 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-Verlag, 2004.

[4] D.J. Abraham, R.W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37:1030–1045, 2007.

[5] D.J. Abraham and T. Kavitha. Dynamic matching markets and voting paths. In *Proceedings of SWAT 2006: the 10th Scandinavian Workshop on Algorithm Theory*, volume 4059 of *Lecture Notes in Computer Science*, pages 65–76. Springer-Verlag, 2006.

[6] D.J. Abraham, A. Levavi, D.F. Manlove, and G. O'Malley. The stable roommates problem with globally-ranked pairs. In *Proceedings of WINE 2007: the 3rd International Workshop on Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, 2007.

[7] D.J. Abraham and D.F. Manlove. Pareto optimality in the roommates problem. Technical Report TR-2004-182, University of Glasgow, Department of Computing Science, 2004.

[8] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the USA*, 43:842–844, 1957.

[9] G. Brassard and P. Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, 1996.

[10] K.S. Chung. On the existence of stable roommate matchings. *Games and Economic Behavior*, 33:206–230, 2000.

[11] M. de Klerk, K. Keizer, W. Weimar, A.J. Goody, A. Spital, and F.L. Delmonico. Donor exchange for renal transplantation. *The New England Journal of Medicine*, 351:935–937, 2004.

[12] E. Diamantoudi, E. Miyagawa, and L. Xue. Random paths to stability in the roommate problem. *Games and Economic Behavior*, 48:18–28, 2004.

[13] T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.

[14] M. Fredman and R.E. Tarjan. Fibonacci heaps and improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

[15] H.N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of STOC'83: the 15th Annual ACM Symposium on Theory of Computing*, pages 448–456. ACM, 1983.

[16] D. Gale. Optimal assignments in an ordered set:an application of matroid theory. *Journal of Combinatorial Theory*, 4:176–180, 1968.

[17] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[18] D. Gale and M. Sotomayor. Some remarks on the stable marriage problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[19] P. Gärdenfors. Assignment problem based on ordinal preferences. *Management Science*, 20:331–340, 1973.

[20] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

[21] M.R. Garey and D.S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[22] D. Gusfield and R.W. Irving. *The Stable Marriage Problem - Structure and Algorithms*. The MIT Press, 1989.

[23] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[24] A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87:293–314, 1979.

[25] R.W. Irving. An efficient algorithm for the stable roommates problem. *Journal of Algorithms*, 6:577–595, 1985.

[26] R.W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.

[27] R.W. Irving. Matching medical students to pairs of hospitals: A new variation on a well-known theme. *Proceedings of ESA'98: the 6th Annual European Symposium on Algorithms.*, 6:381–392, 1998.

[28] R.W. Irving. Greedy and generous matchings via a variant of the Bellman-Ford Algorithm. *Unpublished manuscript*, 2007.

[29] R.W. Irving, T. Kavitha, K. Melhorn, D. Michail, and K. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.

[30] R.W. Irving and D.F. Manlove. The stable roommates problem with ties. *Journal of Algorithms*, 43:85–105, 2002.

[31] R.W. Irving and D.F. Manlove. Some preliminary observations on popular matchings in the roommates problem. *Unpublished manuscript*, 2005.

[32] A. Itai, M. Rodeh, and S.L. Tanimoto. Some matching problems for bipartite graphs. *Journal of the Association for Computer Machinery*, 25(4):517–525, 1978.

[33] T. Kavitha and C. Shah. Efficient algorithms for weighted rank-maximal matchings and related problems. In *Proceedings of ISAAC 2006: the Seventeenth International Symposium on Algorithms and Computation*, volume 4288 of *Lecture Notes in Computer Science*, pages 153–162. Springer, 2006.

[34] D.E. Knuth. *Mariages Stables.* Les Presses de L'Université de Montréal, 1976.

[35] L. Lovász and M.D. Plummer. *Matching Theory.* Number 29 in Annals of Discrete Mathematics. North-Holland, 1986.

[36] M. Lucan, P. Rotariu, D. Neculoiu, and G. Iacob. Kidney exchange program: a viable alternative in countries with low rate of cadaver harvesting. *Transplantation Proceedings*, 35:933–934, 2003.

[37] M. Mahdian. Random popular matchings. In *Proceedings of EC '06: the 7th ACM Conference on Electronic Commerce*, pages 238–242. ACM, 2006.

[38] D.F. Manlove. Personal communication, 2007.

[39] 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.

[40] R.M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of LATIN 2008: the 8th Latin-American Theoretical INformatics symposium,* to appear, Lecture Notes in Computer Science. Springer, 2008.

[41] D. McVitie and L.B. Wilson. The stable marriage problem. *Communications of the ACM*, 14:486–490, 1971.

[42] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. *Unpublished manuscript*, 2006.

[43] J. Mestre. Weighted popular matchings. In *Proceedings of ICALP 2006: the 33rd International Colloquium on Automata, Languages and Programming*, volume 4051 of *Lecture Notes in Computer Science*, pages 715–726. Springer-Verlag, 2006.

[44] 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 in Foundations of Computer Science*, pages 17–27. IEEE Computer Society, 1980.

[45] C. Ng and D.S. Hirshberg. Three-dimensional stable matching problems. *SIAM Journal on Discrete Mathematics*, 4:245–252, 1991.

[46] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity.* Prentice Hall, 1982.

[47] W.R. Pulleyblank. Matchings and extensions. In R.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume 1, chapter 3, pages 179–232. North-Holland, 1995.

[48] E. Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11:285–304, 1990.

[49] A.E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92:991–1016, 1984.

[50] A.E. Roth. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica*, 54:425–427, 1986.

[51] A.E. Roth and A. Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4:131–137, 1977.

[52] A.E. Roth, T. Sönmez, and M. Utku Ünver. A kidney exchange clearinghouse in New England. *The American Economic Review*, 95(2):376–380, 2005.

[53] A.E. Roth, T. Sönmez, and M. Utku Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005.

[54] A.E. Roth, T. Sönmez, and M. Utku Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, May 2004.

[55] A.E. Roth, T. Sönmez, and M. Utku Ünver. Efficient kidney exchange: Coincidence of wants in a structured market. *Unpublished manuscript*, May 2005.

[56] A.E. Roth and M.A.O. Sotomayor. *Two-Sided Matching - A Study in Game-Theoretic Modeling and Analysis*. Cambridge University Press, 1990.

[57] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1:23–37, 1974.

[58] A. Subramanian. A new approach to stable matching problems. *SIAM Journal of Computing*, 23(4):671–700, 1994.

[59] J.J.M. Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12:154–178, 1991.

[60] C-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operational Research*, 23(4):874–891, 1998.

[61] C-P. Teo, J. Sethuraman, and W-P. Tan. Gale-Shapley Stable Marriage Problem Revisited: Strategic Issues and Applications. *Management Science*, 47(9):1252–1267, 2001.

[62] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Proceedings of ISAAC 1997: the 8th Annual International Symposium on Algorithms and Computation*, volume 1350 of *Lecture Notes in Computer Science*, pages 92–101. Springer-Verlag, 1997.