Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Practical applications of one-to-many matchings with one-sided preferences

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Aaron Oertel
geboren am: 18.02.1997
geboren in: Dinslaken

Gutachter/innen: Prof. Dr. Henning Meyerhenke
Prof. Dr. Timo Kehrer

eingereicht am: .... verteidigt am: ....

Contents

1 Abstract

Work in progress

2 Introduction

2.1 Motivation

Many institutions around the world use central, automated matching schemes to assign agents to resources, based on their preferences. For instance, the National Resident Matching Program (NRMP) in the United States uses such a matching mechanism to pair graduating medical students to residency positions at hospitals every year.[1] There are different goals for these matchings, which can range from efficiency to fairness, and finding a mechanism that fulfills these goals is an important task to efficiently design such markets. There are also many variants of matching problems, that include but are not limited to one-to-one, one-to-many and many-to-many matchings. The distribution of preferences need to be considered as well. For instance, in the hospital-residents problem, both the hospitals and residents supply preferences over the other set. Here it's important to consider if incomplete preference lists or ties should be allowed in the matchings.

This thesis will formally describe, examine and analyze the problem of one-to-many matching mechanisms with one-sided preferences. While there are many use-cases for this problem, we will focus on the problem of matching students to seminars, which is of high interest for many universities, who often require their students to participate in a seminar in order to obtain their degree. These students typically have a choice between a handful of different seminars; however, there are capacity constraints that make it hard to match all students to their first choice. Let's consider the following example: 100 students have to be assigned to one of six seminars, wherein each seminar has a capacity of 20 students. The students express their preferences by supplying a strict but incomplete preference list of the six seminars. The goal for the school's administration now is to find an assignment of students to seminars that fulfills their requirements, which can range from assigning as many students as possible to being as fair as possible with the assignments. We will see that there is no obvious choice in picking an algorithm, since there are plenty of trade-offs between existing mechanisms that make it necessary to prioritize the requirements. What makes this problem harder is that the students' preferences are not necessarily equally distributed. Oftentimes a majority of students prefer one seminar which conflicts with other students getting their first choice. At the same time, it can occur that students go unmatched when their preference lists are short and primarily consist of seminars that have already reached full capacity.

The goal of this thesis is to formally model the one-to-many matching problem with one-sided preferences, while presenting different algorithms for finding matchings and finally to evaluate these algorithms using certain metrics that make sense in the one-to-many case. For the purpose of this thesis, we will use the example of student-seminar matching to analyze the problem, but it is important to note that it can just as well be generalized to any other one-to-many matching problem with incomplete preferences. Lastly, an interactive system will be developed, which will allow a school's administration to find a student-seminar matching using one of the presented algorithms.

2.2 Formal Definition

The problem of assigning students to seminars can be described as a many-to-one matching, with a set of students $S := \{s_1, s_2, ..., s_n\}$ and a set of seminars $T := \{t_1, t_2, ..., t_m\}$. Every student $s_i \in S$ provides a strict preference order over a subset of T, and every seminar $t_j \in T$ has a capacity of c_j students. The goal is to find a matching $M : S \rightarrow T$, that assigns students to one of their preferred seminars while respecting the capacity of the seminars. That means that for every seminar t_j the following is satisfied: $|M(t_j)| \leq c_j$. Using this definition we can describe the problem as a many-to-one matching with one-sided, incomplete preferences. We will present different optimality criteria, which can be used for defining an objective function for the problem.

2.3 Related Problems

There are many related matching problems that can be classified as follows:

1. Bipartite matching problems
   a) One-sided preferences
      i. One-to-one (e.g. House Allocation problem)
      ii. **One-to-many (e.g. Capacitated House Allocation problem)**
   b) Two-sided preferences
      i. One-to-one (e.g. Stable-Marriage problem)
      ii. One-to-many (e.g. Hospital-Residents problem)
2. Non-bipartite matching problems
   a) One-to-one (e.g. Stable-Roommates problem)

Even though these problems have some differences, key mechanisms and optimality criteria used for them can nonetheless be similar or even identical. For instance, in the case of one-sided preferences, mechanisms for the one-to-one case can often easily be extended to the one-to-many case. It is also important to note that the problem can be extended to using incomplete preference lists and ties in the preference lists. In fact, real-world settings often necessitate at least incomplete preference lists, as students often want to classify a seminar as unacceptable. The following subsections will briefly present some of the listed matching problems and present their key results.

2.3.1 Stable Marriage Problem

The stable marriage problem was one of the first matching problems to be researched [2], and consequently motivated further research in the field of matching under preferences. The problem is stated as follows: A set of men M and of women W shall be matched one-to-one, where each men and women provide a complete strict-preference order over the agents of the other set. The deferred acceptance algorithm presented in Gale and Shapley's paper[2] finds a stable, complete matching in polynomial time. Stability is defined as follows: given a women $w \in W$ and any man that she was not matched to $m \in M$, w does not prefer m more than her current partner, and m does not prefer w more than his current partner. There are two variants of executing the algorithm, that results in different matchings:

1. the men have priority, by proposing to women, where the woman has to accept the proposal iff it improves her situation.

2. the women have priority and propose to men. This case is analogue to the first one

It has been shown that all possible executions of the algorithm with men as proposers yield the same stable matching. That matching is men-optimal, which means that every man has the best partner that he can have in any stable matching.[3] Additionally, with men proposing, the produced matchings have also been shown to be women-pessimal, meaning that every woman is matched to the worst partner that she can have in any matching.[3] While the scenario of matching women to men like described is presumably not a common real-world occurrence, Gale and Shapley's paper inspired several new approaches to other similar problems:

2.3.2 Hospitals/Residents Problem

A few years after the publication of Gale and Shapley's original paper, it was determined that the deferred acceptance mechanism was essentially the same mechanism used by the National Resident Matching Program (NRMP) in the United States to match graduating medical students to residency positions in hospitals.[3] As a matter of fact, Gale and Shapley's paper described an algorithm for the so-called "college admissions problem"[2], which is essentially the same problem. Just like in the stable-marriage problem, there is a solution that can be hospital-optimal or resident-pessimal.
The problem can be described as finding a one-to-many matching with two-sided, incomplete, but strict preferences. Essentially, a hospital can offer multiple spots and both parties can mark entities from the other set as unacceptable by not including them in their preference lists.[4]
In reality, the problem is a bit more complex, as it permits couples of residents to submit preferences together. It has been shown that a stable solution does not always exist and that finding one if it exists, or showing that it doesn't exist is NP-complete.[5] The revised algorithm used by the NRMP utilizes findings about stability and simple matching markets to find a good approximation, while minimizing opportunities for strategic manipulation, which was indeed possible before.[6]

2.3.3 House allocation problem

Many economists and game theorists[7] have studied variants of the house allocation (HA) problem, wherein a set of indivisible items H needs to be divided among a set A

of applicants. Each applicant may have a strict preference order over a subset of H. Formally this means that an instance I of the problem consists of two disjoint sets, where $H := \{h_1, h_2, ..., h_n\}$ is the set of houses and $A := \{a_1, a_2, ..., a_m\}$ is the set of applicants. Each applicant $a_i \in A$ ranks a subset of the houses in H using a preference list. The houses, on the other hand, do not have any preferences over applicants. A matching or assignment M is a subset of $A \times H$, so that for every applicant a_i, the house $M(a_i)$ is indeed on the applicants preference list.[8]
The house allocation problem is essentially an alias for a matching problem on bipartite graphs with one-sided preferences. There are many applications including matching clients to servers, professors to offices and also students to seminars. For the latter, some generalizations have to be made to the problem; specifically, one seminar should now be matched to more than one student, whereas the houses in HA are matched to one and only one applicant. In the literature, that variant of the problem is also referred to as the "Capacitated House Allocation Problem", denoted by CHA.[9] The following chapters of this thesis, will explore performance indicators and algorithms for finding matchings in the context of students and seminars, which is equivalent to the CHA problem.

2.3.4 Assignment Problem

The problem of matching students to seminars can also be defined as an assignment problem. The goal of the assignment problem is to find a minimum-weight, perfect matching in a bipartite graph. In this case the goal of the problem is, given the set of students S, seminars T, and a cost function $W : S \times T \to I\!R$, to find a map $M : S \to T$, which minimzes the following objective function: $\sum_{s \in S} W(s, M(s))$.
One of the first algorithms used for solving this problem was the Hungarian algorithm by Munkres, which finds a minimum-weight matching in polynomial time.[10]
Alternatively, the problem can be transformed into an instance of the minimum-cost flow problem to determine a minimum weight matching. Section 4.3 will further investigate using this algorithm for the problem of student-seminar matching.

2.3.5 The Student-Project Allocation Problem

A similar problem to the CHA or student-seminar assignment problem is the Student-Project Allocation problem (SPA). This problem considers three entities instead of two, which are students, projects and lecturers. In most variations of the problem, students have preferences over courses, while courses again have capacities. However, lecturers are also considered, where lecturers can have preferences over students and/or courses. Algorithmic approaches used for this problem are mostly based on the deferred acceptance mechanism and find stable matchings that can either favor the students or lecturers.[9]

2.4 Outline

Given this problem definition, we will review the literature for optimality criteria and, based on that criteria, present algorithms that produce matchings with those characteristics. Many of those charactistics are quite obvious such as maximum-cardinality; however, we will also review terms such as "Pareto-efficiency" or "Popularity" that are commonly used in the context of market design and game theory. Using these criteria, we will review matching algorithms that solve the problem and then analyze and compare their runtime-complexity and optimality-properties. We will see that no algorithm is known that fulfills all desirable properties, which makes it necessary to understand the trade-offs and relationships between the algorithms.

After completing a theoretical comparison of the algorithms, we will proceed to analyze matchings computed using real data to better understand the theoretical observations and trade-offs discussed before. Additionally, a web-interface will be developed that allows a university's administration to compute matchings according to their requirements using data that the system receives as input. Lastly, we will briefly explore extensions to the problem, such as the many-to-many case and finally summarize the result and make recommendations for the implementation and usage of the system, based on the current state of research on these matching mechanisms.

3 Optimality criteria

Looking at the previously presented problems, it becomes clear that there are different objective functions or optimality criteria for such matchings. For instance, in the stable marriage problem, matchings are primarily judged by the stability characteristic. However such a characteristic does not make sense in the case of the student-seminar problem, since stability assumes both-sided preferences. For that very reason, different criteria have to be used for judging the quality of a matching.
Intuitively, when thinking about matching students to seminars, it would be desirable to match as many students as possible, as well as matching the students to their first choice, but at the same time to remain fair and resistant to manipulation. To formalize these requirements, a few criteria have been discussed in the literature, which will be helpful for comparing different approaches.

3.1 Maximum cardinality

The goal of the maximum cardinality problem is finding a matching M on a graph $G = (V = (X, Y), E)$, so that $|M|$ is maximal.[11] Consequently, maximum cardinality as an optimality criteria means that a matching is ideal if the number of students that are matched is maximized among all possible matchings. It should be noted here that the students' preferences are not considered when computing the maximum cardinality matching. As a matter of fact, it is possible that multiple matchings of the same cardinality exist, where one of the matchings could be better in the sense of a different optimality criteria. Therefore, maximum cardinality may be desirable, but should be used in conjunction with a different criteria, as it does not consider student preferences. This criteria is also often referred to as "efficiency" in the literature, however we will use the term maximum-cardinality as it is more specific.

3.2 Pareto-Optimality

"Pareto-optimality" or "Pareto-efficiency" is a commonly used term in economics to describe the state of resource allocations. Intuitively an allocation, or in our case a matching, is Pareto optimal iff no improvement can be made to a single individual, without worsening the situation for other individuals. Additionally a matching, in which two students s_1, s_2 would be better off by swapping their seminars is not Pareto optimal. In order to more formally define Pareto-optimality, we must first define student preferences more formally:
Given two matchings M, M' and a student $s \in S$, the student s prefers M' over M in the following cases:

1. s is matched in M' and unmatched in M, or
2. s is matched in both M' and M, however prefers $M'(a)$ over $M(a)$

Using this definition, we now define Pareto optimality as follows: given an instance I of a matching problem and its set of possible matchings $\mathcal{M}$, we define a relation $\succ$

on $\mathcal{M}$, where given two matchings $M, M' \in \mathcal{M}$ the following holds true: $M' \succ M$ if no students prefers M to M', but some student prefers M' over M. Consequently a matching $M' \in \mathcal{M}$ is called Pareto-optimal iff there exists no other matching $M \in \mathcal{M}$, such that $M' \succ M$.[9] A Pareto-optimal matching always exists for any instance of a matching problem and can be efficiently computed using one of various algorithms. A simple greedy algorithm uses the random serial-dictatorship mechanism to draw each agent in random order and lets them select their most-preferred, available item from their preference list.[12, 13] However, the greedy algorithm does not always produce a Pareto-optimal matching of maximum cardinality, which would be desirable in student-seminar matching.[14]

3.3 Popularity

In the context of matching with one-sided preference lists, using an optimality criteria like Stability, which is based on the preferences of both parties, doesn't apply. Instead, a commonly used criteria is a matching M's Popularity, which indicates that more students prefer that matching M over any other possible matching.[15] Given the definition of the student-seminar matching problem, we can formally define Popularity as follows: Let $P(M', M)$ be the set of students who prefer M' over M. A matching M' is said to be more popular than M, denoted by $M' \succ M$, iff $|P(M', M)| > |P(M, M')|$. That concludes that a matching M' is popular, iff there is no other matching M that is more popular than M', i.e. $M' \succ M$.[16, 17] Because of that definition, this criteria is often also referred to as the majority assignment.[18]
Using this definition, we can see that every popular matching also is a pareto-optimal matching. Given a popular matching M' and any matching M for an instance I of the problem, M' is pareto-optimal if $P(M, M') = 0$ and $P(M', M) \geq 1$, which obviously implies that M' is popular as well.[16]
It is important to note that a popular matching's cardinality could be smaller than the maximum cardinality, meaning that, in the case of student-seminar matchings, a group of students could be left unassigned in favor of the majority of the students having a match that they prefer. Additionally, a popular matching, unlike a pareto-optimal matching, does not always exist. To illustrate this, let's consider the following instance: $S = \{s_1, s_2, s_3\}$, $T = \{t_1, t_2, t_3\}$, where each student has the same preference list, being $t_1 < t_2 < t_3$, and each seminar $t_i \in T$ has a capacity of 1. Given the following matchings, we can confirm that there exists no popular matching for the given instance:

1. $M_1 = \{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$
2. $M_2 = \{(s_1, t_3), (s_2, t_1), (s_3, t_2)\}$
3. $M_3 = \{(s_1, t_2), (s_2, t_3), (s_3, t_1)\}$

It is clear that M_2 is more popular than M_1, M_3 is more popular than M_2 and M_1 is more popular than M_3.[17]

3.4 Profile-based optimality

Contrary to Popularity and Pareto-optimality, in which the students' satisfaction with a matching is compared, we should also examine the structure of a matching by defining the profile of a matching and comparing it. Intuitively the profile of a matching M is a vector whose ith component indicates the number of students obtaining their ith-choice seminar in M, according to their preference list.
Formally, let I be an instance and $\mathcal{M}$ the set of its matchings. Given a matching $M \in \mathcal{M}$ with the set of students S and seminars T, we define the regret $r(M)$ of M as follows: $r(M) = \max\{rank(s_i, t_j) : (s_i, t_j) \in M\, s_i \in S, t_j \in T\}$, where for every match $(s_i, t_j) \in M$, $rank(s_i, t_j)$ is defined as the position of t_j on s_i's preference list. The profile of M is now defined as a vector $\langle p_1, ..., p_r* \rangle$, with $r^* = r(M)$ and for each $k \in [1, r^*]$, the kth component is defined as: $p_k = |\{(s_i, t_j) \in M : rank(s_i, t_j) = k\}|$.[9]
Using the defintion of a matching's profile, it is now possible to define a matching as rank-maximal as follows: A matching M is rank-maximal, if its profile $p(M)$ is lexicographically maximum over all possible matchings in $\mathcal{M}$. That means that the number of students in M who are matched to their first choice is maximum among all $M' \in \mathcal{M}$; taking that into consideration, the number of students who are matched to their 2nd choice is maximum among all matchings, and so on. TODO: greedy rank maximal as well?

3.5 Strategy-Proofness

The aforementioned criteria all primarily consider the structure of a matching to evaluate quality and not the properties of a matching mechanism, i.e. algorithm. An important question to consider, however, is if the agents can manipulate the outcome of an algorithm by not truthfully disclosing their preferences - indeed, in the setting of matching residents to hospitals in the US, a previously used algorithm allowed students to improve their outcome of the algorithm by not supplying their real preferences.[3] In the literature, the term for a mechanism, wherein no agent can benefit from misrepresenting their preferences, is called strategy-proof. [16] Such mechanisms are of high interest for most matching problems, since preference-based optimality criteria, like the ones previously mentioned, would certainly lose some significance if the mechanism used to compute them is not strategy-proof. For instance, it has been shown [19] that for the stable marriage problem with incomplete preferences, there exists no matching mechanism that both produces a popular matching and is strategy-proof. To formalize this, we will use a game-theory definition of strategy-proofness, which goes as follows: it is a weakly dominant strategy for each agent to report their true preference list. [16]

3.6 Application to student-seminar matching

Given the problem description of student-seminar matching, it would be desirable to find a matching that has the following properties:

1. **Maximum Cardinality:** As few students as possible should be left unmatched.

2. **Pareto Optimality:** A set of students should not feel the need to swap their match to improve their situation.

3. **Popularity:** The number of students who are satisfied with their matching should be maximum among all possible matchings.

4. **Rank Maximality:** As many students as possible should be matched to their first choice or if not possible, their second choice, and so on.

5. **Stratey-proofness:** Students should not be able to benefit, i.e. increase their chances of being matched to their top-preference, by lying about their true-preferences. A matching mechanism should also not encourage students to supply short preference lists.

Using these requirements, the next sections will present algorithms, for finding matchings that have some of those properties. One of these will be implemented to be used by the interactive system to find matchings that will try to optimize some of the mentioned metrics. As we have already seen, there does not always exists a popular matching or a pareto-optimal matching that is also agent complete. Additionally, the fact that students can supply incomplete preference lists can very well lead to matchings that leave a few students unassigned, even if the sum of the capacity of all seminars is greater than the number of students. Given these constraints and observations, we will see that there is no such thing as an ideal matching for all instances based on the criteria we have presented. However, it will be possible to find matchings that will leave a majority of the students satisfied.

4 Algorithmic Approaches

In the previous section, we discussed several optimality criteria that apply to the problem of matching students to seminars. This chapter will present algorithms for computing matchings that fulfill some of those criteria, as well as evaluating them against each other. The goal of this evaluation is choosing the "ideal" algorithm for implementation. We will see that each of the algorithms has some draw-backs, which might make them undesirable for the student-seminar problem.

4.1 Greedy with serial dictatorship

One of the simplest algorithms for the student-seminar matching problem is a greedy approach, that iterates over the set of students and assigns each of the students to their most preferred seminar that still has some capacity left. In contrast to Gale & Shapley's deferred acceptance algorithm for the stable marriage problem, this algorithm does not tentatively match students once they make their selection, but makes a final assignment for those students. Because of this, the algorithm finds a matching in $\mathcal{O}(n)$ time with n being the number of students. This mechanism of letting students successively pick their highest available preference in order is known as serial dictatorship.[20] In detail the algorithm is as follows:

Algorithm 1 Greedy serial dictatorship matching

Input: set of Students with preferences S, set of Seminars T
Output: Pareto-Optimal Matching M

```
function SD-MATCHING(S, T)
    M = ∅
    for each s ∈ S do
        t = highest ranked, available seminar on preference list of s
        if t ≠ null then
            M = M ∪ {(s, t)}
        end if
    end for
    return M
end function
```

Even though this algorithm is very simple and fast, it has some desirable properties, including one of the optimality criteria defined before:

4.1.1 Properties

Since the order, in which the students get to pick their match is pre-defined, we can easily show that the algorithm always produces a pareto-optimal matching.

Theorem 1. *A greedy algorithm that uses serial dictatorship always produces a pareto-optimal matching.*

Proof. Let M be the matching produced by the algorithm. We assume that there exists a matching N that pareto-dominates M. Now, let $s \in S$ be the first student who prefers his match in N over M. Since s prefers $N(s)$ over $M(s)$, the seminar $N(s)$ must have been unavailable when he made his pick. That means that another student $s' \in S$ exists, who picked $N(s)$ before s could. However, we required that s was matched to a better seminar in N, which means that s' gets a worse match in N. This is a contradiction, so N cannot pareto-dominate M. □

We can also easily see that the algorithm is strategy-proof, because every applicant makes his final pick once it's his turn, there is no benefit in misrepresenting preferences.[16]

4.1.2 Drawbacks

When looking at the algorithm, it is clear that it has a strict preference order over students, specified by the order in which students are matched in the for-loop. Additionally, the algorithm makes no effort to match all students: if it's a student's turn to pick his match, and none of the seminars on his preference lists are free, that student will not be matched at all. This problem gets worse when we consider that our problem statement allows for incomplete preference lists, which increases the chances of having a high number of unmatched students. To illustrate this let us consider the following example in Table 1: In this example, each seminar only has a capacity of 1 and both

Agent	Pref list	Seminar	Capacity
s_1	t_1, t_2	t_1	1
s_2	t_1	t_2	1

Table 1: Instance where Serial Dictatorship admits no max cardinality matching

students have seminar t_1 as their first preference. If the algorithm first gives s_1 a chance to pick, and then s_2, s_1 will be matched to t_1, making t_1 full and not allowing s_2 to be matched. On the other hand, matching s_2 to t_1 first and then matching s_1 to t_1 also yields a pareto optimal matching, however, in this case, all the students are matched.

To address the other problem of preference over students, a simple approach is using the random serial dictatorship mechanism, which instead creates a random order of students as the pick order. This approach is still not fair in the sense that the first student in that order has a better chance at receiving his top priority seminar compared to all other students; however, any student has the chance to be the first one to make a pick.

4.2 Pareto Optimal Maximal Matchings for CHA

We have seen that serial dictatorship is an easy and time-efficient mechanism for computing Pareto-optimal matchings. A big weakness of the approach, however, is that it finds just one of many possible Pareto-optimal matchings without making any guarantees about quality in regards to cardinality. Particularly, the example in Table 1 shows how permutations of the same instance can produce matchings of different cardinality, which motivates the search for an algorithm that produces a Pareto-Optimal matching of maximum cardinality.

(TODO: cited) Abraham et. al [14] have proposed a 3-phase algorithm for computing a maximum cardinality matching for the house allocation problem, which was extended by Sng [8] for the many-to-one case. Before presenting the algorithm, an important lemma about Pareto optimal matchings has to be shown first, which is then used for proofing the correctness of the algorithm. To characterize the lemma, we need to define the terms maximality, trade-in-free and cyclic coalition in regards to a matching M first:

1. **Maximal**: M is maximal, if no student $s_i \in S$ and seminar $t_j \in T$ exists, so that s_i is unassigned, t_j is undersubscribed in M and t_j is on s_i's preference list.[14]

2. **Trade-in-free**: M is trade-in-free, if there are no student $s_i \in S$ and seminars $t_j, t_l \in T$, such that s_i is assigned to t_l, but prefers t_j over t_l and t_j is undersubscribed.[14]

3. **Cyclic coalition**: M contains a cyclic coalition, if there exists a sequence of distinct assigned students $C = \langle s_0, s_1, \ldots, s_{r-1} \rangle$ with $r \geq 2$, such that s_i prefers $M(s_{i+1 \bmod r})$ (i.e. the seminar assigned to the next student in C after s_i) over $M(s_i)$ for every i.[14]

Using these definitions, (TODO) Sng now presents and proofs the following lemma:

Lemma 2. *Let M be a matching of 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.[14]*

Using this lemma, (TODO) Abraham et al [14] construct a 3-phased algorithm, where each phase fulfills one of the properties as described in Lemma 2, like so: Let I be an instance of CHA and G it's underlying graph, then perform the following steps:

1. **Phase 1**: In order to guarantee maximality, compute a maximum matching M in G using Gabow's algorithm. [21]

2. **Phase 2**: Using the matching M produced by step 1, the algorithm now fulfills the trade-in-free criteria as follows: Search for pairs $(s_i, t_j) \in M$ with $s_i \in S$ and $t_j \in T$ and where t_j is undersubscribed in M and s_i prefers t_j over his own match $t_l := M(s_i)$. Whenever such a pair is found, remove the existing assignment (s_i, t_l) and add (s_i, t_j) to M. Consequently t_l is now undersubscribed and may be assigned to another student. Therefore, we continue the search for such pairs until no such pair can be found for every student in S.

3. **Phase 3**: The last phase of the algorithm eliminates any cyclic coalitions from M, if they exist, by using a modified version of Gale's Top Trading Cycles (denoted by TTC) Method.[22] Essentially, the TTC method creates a graph from the

matching M, where every student that is not matched to his most-preferred seminar, denoted by S', is represented by a node. Next, a directed edge is created from each student $s_i \in S'$, to all students in S' who are assigned to the first seminar on s_i's preference list. Now, there must be at least one cycle in this graph, as students may have an edge to themselves. The next step is identifying the cycles and implementing a trade among all agents of that cycle that reassigns the seminars among these students. After the trade, all students from that cycle are removed and these steps are repeated until the graph is empty. Once the graph is empty, M is coalition-free by the correctness of the TTC method.[14]

4.2.1 Properties

Since all modifications to the matching in phase 1 and 2 are limited to swaps and no deletions, maximum cardinality is still guaranteed after the termination of phase 3. Additionally, the resulting matching is also trade-in-free and cyclic-coalition-free as those properties are guaranteed after performing phase 2 and 3 respectively. However, it is important to note that this algorithm, unlike the serial dictatorship mechanism, is not strategy-proof: Due to the fact that a maximum-cardinality matching is computed in step 1, students are encouraged to provide short prefernce lists to have a higher chance of being matched to their first preference.

The runtime of the algorithm is dominated by finding a maximum cardinality matching (phase 1), which yields a time complexity of $\mathcal{O}(E\sqrt{V})$[14] when using the Hopcroft-Karp algorithm. Phase 1 and 2 both take $\mathcal{O}(|E|)$ of time[8], which in total yields a worst-case complexity of $\mathcal{O}(E\sqrt{V})$ for finding a maximum cardinality pareto-optimal matching given any instance I of the problem.

4.3 Assignment Problem

In order to find a rank-maximal matching, we will investigate a set of algorithmic methods that compute a maximum-cardinality, min-weight matching. We can easily see that such a matching must also be rank-maximal, since the min-weight property guarantees that the profile of the matching is lexicographically smallest among all matchings. In section 2.3.4, we briefly presented the assignment problem: it is a combinatorial optimization problem, which assigns a set of agents to a set of tasks, wherein each agent-task tuple is assigned a cost, and to minimize the total cost of the assignment. Formally, we define the problem as follows: Given a set of agents A, tasks T and a map $W(a,t) = w$, with $\forall a \in A, \forall t \in T$, find a bijective map M with: $\forall a \in A, \exists t \in T : M(a) = t$, so that the following objective function is minimized: $\sum_{a \in A} W(a, M(a))$. It is important to note here that the assignment problem tries to find a perfect matching, meaning that all agents are assigned and that a one-to-one matching is found.

One of the first known algorithms for solving this problem is the Hungarian algorithm by Munkres, which finds a perfect, min-weight assignment in $\mathcal{O}(n^4)$ time.[10] However, Jonker et al.[23] have presented a more efficient algorithm, based on Dijkstra's shortest path method, that finds such an assignment in $\mathcal{O}(n^3)$ time. Using these algorithms for the one-to-many case with incomplete preference lists requires a transformation of the input, by introducing artificial edges with large weights, where no edges exist. Additionally, in the case of student-seminar matchings, seminars have to be duplicated according to their capacities to allow for one-to-many matching.

However, we can see that the assignment problem is equivalent to finding a perfect, minimum-weight matching in bipartite, weighted graph. Since the problem of matching students to seminars can simply be described as finding a minimum-weight matching on a bipartite graph, we can construct such a graph, given the set of students, preference lists and seminars and apply graph algorithms that find such matchings for us. In fact, we can simply transform the student-seminar matching problem into an instance of the minimum-cost flow problem. The goal of this algorithm will be to send $|S|$ units of flow through the network, while minimizing the cost of the flow, which is indicated by a seminar's rank on the student's preference lists.

4.3.1 Input Transformation

The problem of matching students as seminars can be given as a bipartite graph $G = (V = (S, T), E)$, where S is the set of students and T the set of seminars. The set of edges E is defined as follows: $E := \{(s, t) \mid s \in S \wedge t \in T \wedge \text{t is on the prefernce list of s}\}$. Additionally, a weight function $W : E \to \mathbb{N}$ is specified, which maps each edge to the position of the seminar on the student's preference list. In order to transform this bipartite graph into an input for the minimum-cost flow problem, a flow network has to be constructed using the bipartite graph. (TODO: define flow network?)

To transform the bipartite graph into a flow network, we first add a source and sink vertex to the graph. Then, we add weights, capacities and edges from and to the source and sink. Specifically, we create an edge from the source to each of the student vertices with a capacity of 1 and a cost of 0. These edges indicate that a student can only be assigned once. Next, for every student we re-use the edges from the bipartite graph G, where each edge $e \in E$ is assigned a capacity of 1 and a cost of $W(e)$. The capacity, again indicates that a student can only be assigned once and the weight indicates the position of the seminar on the student's preference list. Finally, one edge is added from each seminar $s \in S$ to the sink with a capacity of $C(s)$ and a cost of 0.

4.3.2 Properties

The matching M computed by the algorithm is Pareto optimal[8] and has the minimum weight property, and therefore is rank-maximal;[8] however, a large drawback is that this mechanism is not strategy-proof: students are encouraged to provide short preference lists in order to get matched to their most-preferred seminar. The algorithm tries to match every student, which means that students with a list of just one seminar will be prioritized over students, who prefer the same seminar but also supply other preferences. This problem could lead to having all students provide single-element preference lists, which increases the difficulty of finding perfect matchings. Therefore, the algorithm does not encourage the students to supply preference lists that reflect their true preferences, which is a desirable property that has been considered for other matching problems like the hospital-residents-problem.[3]

4.4 Maximum Popular Matchings in CHA

Abraham et al [17] presented an algorithm for finding a popular matching in the house allocation problem (without capacities), which either finds such a popular matching or reports that none exists in $\mathcal{O}(|V| + |E|)$ time. Manlove and Sng [15] extended this algorithm for the many-to-one case, the Capacitated House Allocation problem, by developing a characterization of such popular matchings in CHA and then using it to construct an algorithm that finds a maximum popular matching for any given instance, if it exists. Intuitively, the algorithm will try to match as many applicants as possible to their most-preferred choice to fulfill the popularity criteria. Formally, Manlove and Sng[15] define and proof an alternative characterisation of Popularity in order to develop their algorithm:

4.4.1 An alternative characterization of popular matchings

Given an instance I of the CHA problem, for every student $a_1 \in A$ let $h_j := f(a_1)$ be the first ranked house on s_1's preference list. Furthermore, we call h_j an f-house. For each house $h_j \in H$, define the set of applicants, who named h_j as their first choice, as $f(h_j) = \{a_i \in A : f(a_i) = h_j\}$ and the size of that set as $f_j = |f(h_j)|$. For a matching M in I, we now say that a house $h_j \in H$ is full if $|M(h_j)| = c_j$, i.e. the maximum number of applicants is matched to that house, and undersubscribed if $|M(h_j)| < c_j$. Additionally, for every applicant $a_i \in A$, we append a last-resort house $l(a_i)$ with capacity 1 to a_i's preference list.[15] Manlove and Sng now prove the following lemma [15]:

Lemma 3. *Let M be a popular matching in I. Then for every f-house h_j, $|M(h_j) \cap f(h_j)| = \min\{c_j, f_j\}$.*

In other words, for a popular matching M in I, every f-house h_j is matched to atleast the number of applicants who have h_j as their first-preference but at most h_j's capacity c_j. Next, for every applicant a_i, we define $s(a_i)$ to be the most-preferred house h_j on his preference list, such that either (i) h_j is not an f-house, meaning that it's not the first choice of any applicant, or (ii) h_j is an f-house, but $h_j \neq f(a_i)$ and $f_j < c_j$. In simple terms, $s(a_i)$ is the first undersubscribed house on a_i's preference list after $f(a_i)$. We will refer to such houses as s-houses. It is important to note that such a house $s(a_i)$ always exists, due to the introduction of $l(a_i)$. In a popular matching, an agent a_i may only be matched to either $f(a_i)$ or $s(a_i)$, as every house in between those two is full according to the defintion of $f(a_i)$ and $s(a_i)$. Manlove and Sng, again prove the following two lemmas [15]:

Lemma 4. *Let M be a popular matching in I. Then no agent $a_i \in A$ can be matched in M to a house between $f(a_i)$ and $s(a_i)$ on a_i's preference list.*

Lemma 5. *Let M be a popular matching in I. Then no agent $a_i \in A$ can be matched in M to a house worse than $s(a_i)$ on a_i's preference list.*

To summarize, so far for every applicant $a_i \in A$ we have defined the applicant's most preferred house $f(a_i)$ and his second most-preferred, but available house $s(a_j)$. We have seen that, in a stable matching, applicants can only be matched to either of those houses $f(a_i)$ or $s_(a_i)$. Using this information, we can now construct a subgraph G' of $G = (V = (A, H), E)$, by removing all edges in G from every applicant a_i, except the ones to $f(a_i)$ and $s(a_i)$. We now say that a matching M is agent-complete in G' if it matches all agents in A and no agent a_i is matched to their last-resort house $l(a_i)$.[15] Manlove and Sng prove the following theorem to fully characterize popular matchings[15]:

Theorem 6. *A matching M is popular in I iff:*

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'*

4.4.2 Algorithm

Using Theorem 6, Manlove and Sng develop the algorithm Popular-CHA for findin a maximum popular matching or reporting that none exists.[15] The algorithm works as follows:

1. Reduce G to G'.

2. Match all agents to their first-choice house h_j, if $f_j \leq c_j$, i.e. the house would be undersubscribed or just full afterwards. This will satisfy condition 1a of Theorem 6.

3. Remove all applicants and their incident edges, that were matched in the previous step, from G'. Additionally, update the capacities for each previously matched house h_j as $c'_j = c_j - f_j$. All full and isolated houses and their incident edges are also removed from G'.

4. Compute a maximum cardinality matching M' on G' using the updated capacities. For this step Manlove and Sng use Gabow's algorithm [21].

5. If M' is not agent complete, then no popular matchings exists. Otherwise merge the matchings M and M'.

6. As a last step, to fulfill condition 1b of Theorem 6, promote any agent $a_i \in M$ who is matched to their s-house to their f-house, if it's undersubscribed.

4.4.3 Properties

Due to the fact that step 2, 4 and 6 make the computed matching fulfill the criteria outlined in Theorem 6, the algorithm produces a popular matching of maximum cardinality, if it exists. Furthermore, its runtime complexity is $\mathcal{O}(\sqrt{C}n_1 + |E|)$, where C is the sum of the capacities of the houses and n_1 the number of applicants. $|E|$ is equivalent to the sum of the agents' preference list lengths. The runtime is dominated by Gabow's algorithm, which computes the maximum cardinality matching in G' in $\mathcal{O}(\sqrt{C}n_1)$.[15] Alternatively, a modified version of the Hopcroft-Karp algorithm could be used for computing the maximum cardinality matching in $\mathcal{O}(E\sqrt{V})$[24] time, which is what Abraham et al. use for the one-to-one case.

5 Comparison of mechanisms

In the previous chapter we have studied several different algorithmic approaches, which guarantee different optimality criteria and rely on different mechanisms. To better evaluate the algorithms against the optimality criteria we defined in chapter 3 we will now summarize and compare the properties of the algorithms. Afterwards we will look at some practical results, that were obtained by a benchmark of matching mechanisms with one- and two-sided preferences by Diebold and Bichler.[25] These results will provide a guideline in picking algorithms for implementation and will be described in the next chapter.

5.1 Theoretical results

Drawing back to the list of desirable properties defined in section 3.6, let us now recap and compare the aforementioned algorithms to evaluate which one could be applicable for the problem of matching students to seminars. Unfortunately none of the algorithms guarantee all of the optimality critera at the same time, which makes the choice of an algorithm unclear. Table 2 gives an overview of the presented algorithms and their properties. Each of the algorithms is listed in the same order that they were presented, and for each optimality criteria, a yes/no encoding is used to make a statement about which properties an algorithm guarantees. It is important to note here that a "no" in a column does not strictly mean that the given optimality criteria cannot be fulfilled by the algorithm, but rather that the algorithm does not guarantee it. For instance, a matching computed with the greedy algorithm can be of maximum cardinality or be popular. Only the results for strategy-proofness are a strict yes or no, since fulfilling strategy-proofness does not depend on the instance of the problem, but only of the mechanism being used.

	Greedy	Max Pareto	Assignment	Popular
Maximum Cardinality	no	yes	yes	yes
Pareto-Optimal	yes	yes	yes	yes
Popular	no	no	no	yes
Rank Maximal	no	no	yes	no
Always Exists	yes	yes	yes	no
Strategy Proof	yes	no	no	yes
Time Complexity	$\mathcal{O}(n)$	$\mathcal{O}(\sqrt{n}m)$	$\approx \mathcal{O}(n^3)$	$\mathcal{O}(\sqrt{C}n_1 + m)$

Table 2: Comparison of different algorithmic approaches

To summarize the results, we can see that all of the algorithms guarantee pareto-optimality, however only the Popular-CHA algorithm guarantees popularity. At the same time, only the greedy approach and Popular-CHA also guarantee strategy-proofness, which makes Popular-CHA particularly interesting for the student-seminar problem.

5.1.1 Strategy-proofness and maximum cardinality

One interesting observation is that fulfilling maximum cardinality comes at the cost of either not being strategy-proof, or not guaranteeing that a matching exists at all. Indeed, only the greedy and Popular-CHA algorithm guarantee strategy-proofness. However, ensuring strategy-proofness and maximum cardinality at the same time comes at the cost of not always finding a matching. If we look back at the algorithm Popular CHA, we remember that a maximum cardinality matching M' is computed on the reduced graph G'. We saw that a maximum popular matching does not exist, iff the matching is not agent-complete, meaning that one of the agents is matched to their last-resort house. While this mechanism ensures strategy-proofness, it is also not always possible to find such a maximum cardinality matching using the Popular CHA algorithm. Therefore, it remains an open question whether or not a mechanism exists that both is strategy-proof and produces maximum-cardinality matchings.

5.1.2 Max-PaCHA and the assignment problem

Another important thing to notice is the similarity of the properties between the Max-PaCHA and assignment problem algorithm. Except for the fact that the assignment algorithm guarantees rank maximality, the two algorithms produce matchings with very similar characteristics, which then begs the question why one should use the Max-PaCHA algorithm. But looking at the runtime complexity of the algorithms, we see that, while both algorithms run in polynomial time, the assignment problem takes longer to be solved.

5.2 Practical results

In the past, the technical university of Munich (TUM) assigned students to courses via a first-come first-served mechanism, which resulted in complaints from both students and lecturers.[26] After Diebold et al. [27] at TUM investigated different mechanisms for this matching problem, the university switched to a stable matching mechanism, which is based on Gale & Shapley's deferred acceptance mechanism. It has to be noted here, that the university uses two-sided preferences for their system, which makes their results less applicable for this thesis, however Diebold et al. have also published an extensive benchmark on matching mechanisms with both one- and two-sided preferences.[25] They used real course registration data from TUM, grouped into 28 datasets for investigating properties, including size, rank and popularity, of matchings produced by several mechanisms.

5.2.1 Dataset

The data with one-sided preferences is comprised of 9 datasets from the official tutorial registration at TUM in the period between October 2012 and October 2015. All of those datasets contain incomplete preference lists and all but two of them contain ties. Each dataset contains between 136 and 1035 students and between 5 and 51 courses with total capacities ranging from 130 to 1282. For the dataset with 51 courses, the authors provide a histogram of the length of student's preference lists, showing that a majority of students gave preferences for between 10 and 15 tutorials.[25] Unfortunately the rest of the data is kept private and therefore it's not possible to get information on the distribution of the indiviudal preferences lists, or in other words the popularity of each seminar from the students' perspective.

5.2.2 Algorithms used

Diebold et al. used most of the mechanism described in section 4, with some small differences. We will primarily compare their results for the following algorithms:

1. RSD - Random Serial Dictatorship, see 4.1

2. MPO CHA - Max-Pareto-Optimal CHA, see 4.2

3. ProB CHAT - Profile-based optimal algorithm with ties. Produces the same matching as the assignment problem algorithm, see 4.3

4. Pop CHAT - Max Popular Algorithm with ties. An extension of Pop-CHA, see 4.4

5.2.3 Methodology

The authors implemented all matching mechanisms in Python 3.4.2 and used randomization to break ties. Additionally, every algorithm was run 100 times and the average of the metrics were obtained for each mechanism. Additionally some of the following metrics were used for the one-sided case:

1. **Size**: Number of students being matched.

2. **Average Rank**: A rank for a student is the position of the student's assigned course on his preference list. The average is taken over all students for this metric.

3. **Popularity**: The mechanisms are compared against each other to check which mechanism produces more popular matchings.

4. Average **AUPCR** (Area under the Profile Curve Ratio): This metric was introduced to compare rank profiles of different matchings. In the words of the authors: "The AUPCR up to a specific rank describes the probability that a matching mechanism will rank a randomly chosen student higher than his n-th preference."[25]

5.2.4 Results

A summary of the benchmark is provided in Table 3. Unsurprisingly the Pop CHAT mechanism produces more popular matchings than all other mechanisms, but it's also interesting to note that ProB CHAT produces more popular matchings than all mechanisms but Pop CHAT. Besides that we can see that all mechanisms achieve an average size of atleast 97.4%, while MPO CHA and ProB CHAT unsurprisingly find perfect matchings.

Metric	RSD	MPO CHA	ProB CHAT	Pop CHAT
Average AUPCR	94.95%	96.77%	97.83%	97.31%
More Popular	20%	0%	80%	100%
Average rank	1.41	1.51	1.26	1.33
Average size	97.48%	100%	100%	99.78%
Max Runtime	0.014s	0.522s	33.852s	2.458s

Table 3: Summary of one-sided matching mechanisms from Diebold et al. [25]

Looking at the profile and rank, we can see that ProB CHAT performs best, which is not a surprise, but at the same time Pop CHAT got very close in terms of average AUPCR and average rank. However it's important to note that Pop CHAT didn't find a matching for one of the 9 datasets. At first glance it would seem that ProB CHAT always produces really good results in terms of the metrics used, however we also need to note that it's maximum runtime in the benchmark was 33.852 seconds for the largest dataset with 915 students and 51 courses with a total capacity of 1080. In contrast, the RSD mechanism still produces acceptable results at runtimes of a fraction of a second, while being strategy-proof.

5.2.5 Learnings

In this benchmark, we have seen that the ProB CHAT mechanism unsurprisingly produces some of the best results, while also being limited by its runtime which is worse among all mechanisms. Additionally, we have seen that Pop CHAT also produces good matchings at a more acceptable runtime. One interesting observation is that Pop CHAT produces better results than MPO CHA, if it can find a matching. While these results confirm some of the theoretical observations, it will be interesting to get more insights with different data being used. In this benchmark, most of the data contained ties and we didn't get any insights on the distribution of preferences.

6 Implementation

6.0.1 Overview

To allow for further experimentation with my findings, I will describe a prototype for an interactive web-system, built for computing student-seminar matchings. The requirements for this system were quite simple to keep it in scope of this thesis. The system constists of a website, which communicates with a server that manages the underlying data and computes the matchings. The requirements can be listed as follows:

1. Allow for adding and deleting student and seminar data.
2. Allow for computing a match using the previously entered data.
3. Output the matching and it's properties.

6.0.2 Algorithm Implementation

6.0.3 Web-Interface Implementation

7 Evaluation

8 Extensions to the problem

For the bulk of this thesis, we have looked at many-to-one matching problems with one-sided preferences, as this settings makes the most sense for the student-seminar application. However, there is a wide range of similar problems and extensions that are also worth mentioning. This section will present some of those problems and key results from the literature.

8.1 Two-Sided preferences

8.2 Many-to-Many matchings

8.3 Online variant

When solving the online-variant of the problem, the whole input is not available from the start. That means that the input needs to be processed piece by piece, or more formally: Given a bipartite weighted graph (U, V, E), where U is known to the algorithm, vertices in V are unknown, but arrive one at a time, while also revealing their incident edges, find a matching that maximizes some objective function. These algorithms could be of interest in the case of a first-come first-serve course allocation system, or in other areas such as DVD-rental or online-advertisement allocation systems.[28]

In the case of student-seminar assignments, we would assume that the set of courses and their capacities is known beforehand, and the students arrive later. One of the algorithms we have seen in section 4 can be used for this problem, namely the RSD-algorithm. As a matter of fact, the algorithm will produce the same results for the offline and online case, given that the order in which students are processed is identical.

8.3.1 Online max-cardinality matching

There has been lots of research in particular on finding maximum-cardinality matchings with online inputs. The online-inputs are classified by how much information the algorithm possesses about the input order. For now, we will only consider the **Adversarial order**, where we assume no knowledge of the query sequence, which means that only U is known at the beginning of the algorithm, while we have no knowledge of V and E or the order they appear in.[28] To measure performance we will use the **competetive ratio** of an algorithm which is defined as follows. Given an instance of the problem I, the value of the objective function for the online algorithm is given as $ALG(I)$, and the value of the objective function for the best offline algorithm is given as $OPT(I)$. The competetive ratio is now computed as follows: $C.R. = \frac{ALG(I)}{OPT(I)}$.[28]

Simple algorithms for this online problem, are a greedy algorithm, that matches arriving vertices to any available neighbor, or a random approach that matches arriving vertices to a random neighbor. These mechanisms achieve a competetive ratio of $\frac{1}{2}$.[28] An optimal, but yet simple algorithm was introduced by Karp et al.[29], which achieves a competetive ratio of $1 - \frac{1}{e} \simeq 0.63$. The algorithm, called Ranking, begins by permutating the known vertices of U in a random permutation π, i.e. we assign a random priority number to each $u \in U$. Each incoming vertex $v \in V$ is then assigned to an available neighbor, with the smallest value of $\pi(u)$. In detail the algorithm looks like this:

Algorithm 2 Ranking

```
Offline: Pick a random, uniform permutation π of U
for each arriving vertex v ∈ V do
    if v has no available neighbors then
        continue
    end if
    Match v to the neighbor u ∈ U with the smallest value π(u)
end for
```

9 Conclusion

References

[1] A. E. Roth, "The evolution of the labor market for medical interns and residents: A case study in game theory," *Journal of Political Economy*, vol. 92, no. 6, pp. 991–1016, 1984.

[2] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[3] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*. Cambridge, MA, USA: MIT Press, 1989.

[4] A. Roth, "The theory and practice of market design," Nobel Prize in Economics documents 2012-5, Nobel Prize Committee, 2012.

[5] E. Ronn, "Np-complete stable matching problems," *Journal of Algorithms*, vol. 11, no. 2, pp. 285 – 304, 1990.

[6] A. E. Roth and E. Peranson, "The redesign of the matching market for american physicians: Some engineering aspects of economic design," Working Paper 6963, National Bureau of Economic Research, February 1999.

[7] S. P. Fekete, M. Skutella, and G. J. Woeginger, "The complexity of economic equilibria for house allocation markets," *Information Processing Letters*, vol. 88, no. 5, pp. 219 – 223, 2003.

[8] C. Thiam Soon Sng, *Efficient Algorithms for Bipartite Matching Problems with Preferences*. PhD thesis, University of Glasgow, 7 2008. A thesis submitted to the Faculty of Information and Mathematical Sciences at the University of Glasgow for the degree of Doctor of Philosophy.

[9] D. F. Manlove, *Algorithmics of Matching Under Preferences*, vol. 2 of *Series on Theoretical Computer Science*. WorldScientific, 2013.

[10] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[11] D. B. West, *Introduction to Graph Theory (2nd Edition)*. Pearson, 2000.

[12] A. E. Roth and M. A. O. Sotomayor, *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Econometric Society Monographs, Cambridge University Press, 1990.

[13] A. Abdulkadiroglu and T. Sonmez, "Random Serial Dictatorship and the Core from Random Endowments in House Allocation Problems," *Econometrica*, vol. 66, pp. 689–702, May 1998.

[14] D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn, “Pareto optimality in house allocation problems,” in *Proceedings of the 16th International Conference on Algorithms and Computation*, ISAAC’05, (Berlin, Heidelberg), pp. 1163–1175, Springer-Verlag, 2005.

[15] D. Manlove and C. Sng, “Popular matchings in the capacitated house allocation problem,” September 2006.

[16] B. Klaus, D. F. Manlove, and F. Rossi, “Matching under preferences,” in *Handbook of Computational Social Choice* (F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, eds.), pp. 333–355, Cambridge ; New York: Cambridge University Press, April 2016.

[17] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, “Popular matchings,” in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’05, (Philadelphia, PA, USA), pp. 424–432, Society for Industrial and Applied Mathematics, 2005.

[18] P. Gärdenfors, “Match making: Assignments based on bilateral preferences,” *Behavioral Science*, vol. 20, no. 3, pp. 166–173, 1975.

[19] A. E. Roth, “Incentive compatibility in a market with indivisible goods,” *Economics Letters*, vol. 9, no. 2, pp. 127 – 132, 1982.

[20] M. Manea, “Serial dictatorship and pareto optimality,” *Games and Economic Behavior*, vol. 61, no. 2, pp. 316 – 330, 2007.

[21] H. N. Gabow, “An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems,” in *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC ’83, (New York, NY, USA), pp. 448–456, ACM, 1983.

[22] L. Shapley and H. Scarf, “On cores and indivisibility,” *Journal of Mathematical Economics*, vol. 1, no. 1, pp. 23 – 37, 1974.

[23] R. Jonker and A. Volgenant, “A shortest augmenting path algorithm for dense and sparse linear assignment problems,” *Computing*, vol. 38, pp. 325–340, Dec 1987.

[24] J. E. Hopcroft and R. M. Karp, “A n5/2 algorithm for maximum matchings in bipartite graphs,” in *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pp. 122–125, Oct 1971.

[25] F. Diebold and M. Bichler, “Matching with indifferences: A comparison of algorithms in the context of course allocation,” *European Journal of Operational Research*, vol. 260, no. 1, pp. 268 – 282, 2017.

[26] "Course Assignment at the Department of Informatics." `https://matching.in.tum.de/`, 2019. [Online; accessed 09-June-2019].

[27] F. Diebold, H. Aziz, M. Bichler, F. Matthes, and A. Schneider, "Course allocation via stable matching," *Business & Information Systems Engineering*, vol. 6, pp. 97–110, Apr 2014.

[28] A. Mehta, "Online matching and ad allocation," *Found. Trends Theor. Comput. Sci.*, vol. 8, pp. 265–368, Oct. 2013.

[29] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, (New York, NY, USA), pp. 352–358, ACM, 1990.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den June 10, 2019 ....