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 universities require students to enroll in a seminar in order to obtain their degree. Usually the students have a choice between a handful of different seminars, however there are capacity constraints that make it hard to give all students their first choice. Let's consider the following example: 100 students have to be assigned to one of 6 seminars, where each of the seminars has a capacity of 20. The students express their preferences by supplying a strict, but incomplete preference list of the 6 seminars. The goal for the school's administration now is to assign as many students as possible to a seminar of their choice.
What makes this problem harder is that the students preferences aren't necessarily equally distributed. Oftentimes a majority of students prefers one seminar in which case conflicts exist in choosing which students get their first choice. At the same time, it can happen that students go unmatched if their preference lists are short and full of seminars that have reached full capacity already.
The goal of this thesis is to formally model the aforementioned problem, while presenting different algorithms for finding possible matchings and finally to evaluate these algorithms using certain metrics that make sense in the domain of matching, such as stability, rank-maximality, popularity and pareto-optimality. Additionally, an interactive system will be developed, which allows 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 $M : S \to T$, that assings students to 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.

2.3 Related Problems

There are many related problems, which differ in the preference lists' structure and by how many entities of the first set are matched to how many entities of the second set. Studying these problems reveals some important insights into matching markets.

2.3.1 Stable Marriage Problem

The stable marriage problem was one of the first matching problems to be researched[1] and consequently motivated a lot more research in the domain of matching.

The problem is stated as follows: A set of m men and n women 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[1] finds a stable, complete matching in polynomial time. Stability is defined as follows: given a women w and any man that she was not matched to m, w does not prefer m more than her current partner, and m does not prefer w more than his current partner.
TODO: present algorithm here?
The algorithm can be executed in two ways:

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.[2] Additionally, with men proposing the produced matchings has also been shown to be women-pessimal, meaning that every woman is matched to the worst partner that she can have in any matching.[2]
While the scenario of matching women to men like described (hopefully) doesn't occur in the real world, Gale and Shapley's paper inspired a lot more approaches to other practical problems:

2.3.2 Hospitals/Residents Problem

A few years after the publication of Gale and Shapley's original paper, it was found that the deferred acceptance algorithm was essentially the same algorithm used by the National Resident Matching Program (NRMP) in the United States to match graduating medical students to residency positions in hospitals.[2] As a matter of fact, Gale and Shapley's paper described an algorithm for the so-called "college admissions problem"[1], which is essentially the same problem. Just like in the stable-marriage problem, there is a solution that 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 list.[3]
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.[4]
The revised algorithm used by the NRMP uses findings about stability and simple matching markets to find a good approximation, while minimizing opportunities for strategic manipulation, which was indeed possible before.[5]

2.3.3 House allocation problem

Many economists and game theorists[6] have studied variants of the house allocation (HA) problem, where 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.[7]
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 generalisations 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.[8] (TODO)
The next section will explore performance indicators and algorithms for finding such matchings in the context of student-seminar matchings.

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 A, seminars B, and a cost function $W : A \times B \to IR$, to find a map $M : A \to B$, which minimzes the following objective function: $\sum_{a \in A} W(a, M(a))$.
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.[9]
Alternatively the problem can be transformed into an instance of the minimum-cost flow problem to determine a minimum weight matching. Section 3.4 (TODO) will further investigate using this algorithm for the problem of student-seminar matchings.

2.4 Outline

Now that the problem has been formalized and similar problems have been presented, I will present several different algorithmic approaches to find possible matchings. To evaluate these matchings I will also present commonly used metrics like:

- Rank-maximality (maximum number of students ranked to their first priority)
- Maximum cardinality
- Pareto optimality
- Popularity

- Profile-based optimality

Using these metrics I will choose one algorithm for implementation and analyze it's space and time complexities as well as evaluating it's performance using the aforementioned metrics against the other approaches. Next, I will describe the interactive web system developed for using the algorithm and lastly extend the problem to a two-sided problem.

3 Optimality criteria

Looking at the previously presented problems, it is 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, etc. 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.[10] 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 student's 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.

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 defintion 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 $M' \succ M$.[8] 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 let's them select their most-preferred, available item from their preference list.[11, 12] However the greedy algorithm does not always compute a Pareto optimal matching of maximum cardinality, which would be desirable in student-seminar matching.[13]

3.3 Popularity

In the context of matching with one-sided preference lists, using an optimality criteria (TODO) like Stability, that 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.[14] 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. $M' \succ M$.[15, 16] Due to that definition, this criteria is often also referred to as the majority assignment.[17]
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.[15]
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. (TODO: sentence) 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. $M_3 = \{(s_1, t_2), (s_2, t_3), (s_3, t_1)\}$

Obviously 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.[16]

3.4 Profile-based optimality

Contrary to Popularity and Pareto optimality, where the students' satisfaction with a matching is compared, we could 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: $p_k = |\{(s_i, t_j) \in M : rank(s_i, t_j) = k\}|$.[8]
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 $M' \in \mathcal{M}$ and 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 previously mentioned criteria all have in common that they 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 though, 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 the outcome of the algorithm by not supplying their real preferences.[2] In the literature, the term for a mechanism, where no agent can benefit from misrepresenting their preferences is called strategy-proof. [15] 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 [18], 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. [15]

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 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 mechansim 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 which 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 have 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. Due to that this 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.[19] In detail the algorithm looks like this:

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 of the computed matching

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. □

TODO: citation

We can also easily see that the algorithm is strategy-proof. [15] Because every applicant makes his final pick once it's his turn, there is no benefit in misrepresenting preferences.

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's 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, than 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 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 [13] have proposed a 3-phase algorithm for computing a maximum cardinality matching for the house allocation problem, which was extended by Sng [7] 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.[13]

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.[13]

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.[13]

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.[13]*

Using this lemma, (TODO) Abraham et al [13] 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. [20]

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.[21] 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 atleast 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.[13]

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. The runtime of the algorithm is dominated by finding a maximum cardinality matching (phase 1), which yields a time complexity of $\mathcal{O}(E\sqrt{V})$[13] when using the Hopcroft-Karp algorithm. Phase 1 and 2 both take $\mathcal{O}(|E|)$ of time[7], 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 have briefly presented the assignment problem. To recap, the assignment problem is a combinatorial optimization problem, which assigns a set of agents to a set of tasks, where 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 algorithms for solving this problem is the Hungarian-algorithm by Munkres, which finds a perfect, min-weight assignment in $\mathcal{O}(n^4)$ time.[9] However, Jonker et al.[22] 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 easily 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.
Indeed, 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. Specificially, 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. Last, 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 of the computed matching

The matching M computed by the algorithm is Pareto optimal[7] and has the minimum weight property, and therefore is rank-maximal, however a big drawback is that the algorithm is not strategy-proof: Using this mechanism, 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 prioitized over students, who also prefer that seminar but also supply other preferences. This problem could lead to having all students supply singleton preference lists, which also makes it harder to find 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.[2]

4.4 Maximum Popular Matchings in CHA

Abraham et al [16] 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 [14] extended this algorithm for the many-to-one case, the capacitated house allocation problem, by developing a characterisation 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[14] 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.[14] Manlove and Sng now prove the following lemma [14]:

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 [14]:

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 $l(a_i)$.[14] Manlove and Sng proof the following theorem to fully characterize popular matchings[14]:

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.[14] 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 [20].
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 it's 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 lenghts. The runtime is dominated by Gabow's algorithm, which computes the maximum cardinality matching in G' in $\mathcal{O}(\sqrt{C}n_1)$.[14] Alternatively, a modified version of the Hopcroft-Karp algorithm could be used for computing the maximum cardinality matching in $\mathcal{O}(E\sqrt{V})$[23] time, which is what Abraham et al. use for the one-to-one case.

4.5 Comparison of theoretical results

Drawing back to the list of desirable properties defined in section 3.6 we should now recap and compare the mentioned 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 not obvious. 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 in 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 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.

4.5.1 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 is an open question whether a mechanism exists that both is strategy-proof and produces maximum-cardinality matchings.

4.5.2 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 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 Implementation

test

6 Evaluation

test

7 Extensions of the problem

7.0.1 Two-Sided Preferences

7.0.2 Many-to-Many matchings

8 Conclusion

References

[1] 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.

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

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

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] D. Manlove and C. Sng, "Popular matchings in the capacitated house allocation problem," September 2006.

[15] 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.

[16] 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.

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

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

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

[20] 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.

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

[22] 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.

[23] 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.

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 5, 2019 .....