V nedávné době přišli Steindl Zehavi s novým pohledem na modelování složitějších preferenčních relací agentů v přiřazování. Úkolem je přiřadit nanejvýš jeden předmět každému agentovi tak, že výsledné přiřazení bude závidění prosté ve všech vrstvách agentských preferencí. Jak se ukazuje, pro většinu zadání taková přiřazení nemohou existovat a je algoritmicky velice obtížné je nalezt.

Cílem práce je nastudovat dostupnou literaturu a následně prozkoumat možnosti relaxování podmínek původně definovaného přiřazení. Dále bude třeba nalezt nové koncepty tzv. férovovosti přiřazení a prozkoumat algoritmickou složitost navrhovaných konceptů. Zároveň by bylo dobré zjistit strukturální vlastnosti instancí, které garantují nějakou férovost.

Bachelor's thesis

NEW MODELS IN ASSIGNMENT UNDER MULTIPLE PREFERENCES

Kryštof Razima

Faculty of Information Technology
Katedra teoretické informatiky
Supervisor: doc. RNDr. Dušan Knop, Ph.D.
May 6, 2022

Contents

I am dearly obliged to doc. RNDr. Dušan Knop, Ph.D. for giving me an opportunity to work on this thesis and that he has provided valuable information about Theoretical Informatics.

Furthermore as this is my first thesis, I should acknowledge all the feedback and insights about scientific writing. Many consulting took place and without it following work would not have been possible.

Thank you.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act

In Praze on May 6, 2022 .....

Abstrakt

Přiřazovací problém je problémem na pomezí matematické a ekonomické teorie. Přiřazovací problém se skládá z agentů a položek, agenti vyjadřují své preference k položkám a cílem je najít (určitým způsobem) optimální přiřazení. Problém přiřazení je velmi dynamický problém, protože může být vzato v úvahu mnoho dalších aspektů. V tomto příkladu mají agenti více seznamů preferencí, jak je předesláno v *Parameterized Analysis of Assignment Under Multiple Preferences*. My jsme však k problému přistoupili jinak než v článku, motivováni množstvím výzkumů, které již byly provedeny v oblasti problému přiřazení jednoho preferenčního profilu, uvádíme syntézu jednoho profilu z více profilů. Stejně jako představení algoritmu pro nalezení Pareto optimálního řešení na takových profilech. Vzhledem k tomu, že námi prezentované profily umožňují rovnost mezi dvěma položkami (což znamená, že žádná z nich není preferována před druhou), museli jsme zvolit poněkud odlišný přístup. Hlavním cílem bylo najít algoritmické řešení se složitostí alespoň v polynomiálním čase, což se podařilo. Zabýváme se také různými aspekty spravedlnosti při takovém zadání. Následující řešení by měla být přínosná nejen pro Přiřazovací problém s více preferencemi, neboť představují nový pohled na nedávno publikovaný problém, ale také pro Přiřazovací problém s nestriktními profily.

Klíčová slova Přiřazovací problém, Pareto optimalita, Přiřazovací problém s nestriktními profily, Přiřazovací problém s více preferencemi

Abstract

The assignment problem is a problem on intersection of mathematical and economical theory. Assignment problem consist of agents and items, the agents express their preferences over items and the goal is to find (in some way) optimal allocation. The assignment problem is very dynamical problem, because lot of others aspects might be taken into consideration. In this example agents have multiple preference lists as presneted in *Parameterized Analysis of Assignment Under Multiple Preferences*. However we took different approach to the problem than in the article, motivated by the amount of research already done on single preference profile assignment problem, we present synthesising one profile from multiple ones. As well as presenting an algorithm for finding Pareto optimal solution on such profiles. Since the profiles we presented allowed equations between two items, (meaning none of them is preferred over the other) we had to take a slightly different approach. The main focus was to find algorithmic solution with at least polynomial time complexity, which was achieved. We also discuss different aspects of fairness in such assignments. following solutions should be beneficial not only to Assignment Under Multiple Preference as to present a new view on the recently published problem, but also to the Assignment problem with non-strict profiles.

Keywords Assignment problem, Pareto Optimality, Assignment problem with non-strict profiles, Assignment Under Multiple Preference

Introduction

Let us start with an example. Three brothers encounter a magical jhin who has three gifts that he will grant the brothers, unlimited power, invisibility, and immortality. However, each of them can only take one of the three gifts mentioned. The brothers must come to an agreement on who will take which gift. Or there should be some way to distribute those things fairly among all participants. But fairness is not something simple and there are multiple ways we can define fairness. For example, brothers could choose one wish at a time. In order from the oldest to the young, or maybe from the youngest to the oldest, or the order in which they choose could be random (e.g., the outcome of rock-paper-scissors, by lottery...). All of those solutions would be fair in some sense. What we have informally described here are some cases of serial dictatorship. Which is a method commonly used for such problems [1]. Serial dictatorship offers some fairness namely, Pareto optimality, which means that there are not two or more people who would be willing to swap their item for another one so that they would all benefit if they traded with each other. There may be some that are not happy with their item, but they would simply not be able to find anyone willing to trade with them. So we can say that if someone has an item I would like more than my own, that he prefers his item over mine. As this is some defined fairness, however, there may be some who will be much worse off than others. For such cases, we might want results that are envy-free, meaning that no one would even be wanting to change their item for someone else, which is another common method of fairness[2]. In such an example, there can be observed some similarities to the cake cutting problem [3]. Where also some mechanics and definitions of fairness must be considered.

If we extract this to abstract and less ordinary terms. We have a set of agents and a set of items. There might be more items than agents or vice versa. Every agent can only take one item and one item can be taken by only one agent. Each agent has some preferences over the items. The dilemma of the problem is which agent will get what item. That would be an instance of an assignment problem, of which there are many versions. The solution is usually called an assignment, where the assignment states which agent will get what item. This problem is usually compared to a campus problem where each student prefers different housing units[4]. However, items don't have to be always a positive thing, that people only benefit from having, it can be distributions of housework or tasks in a company.

There can be other components of the assignment problem as well [5]. Ownership could be part of the problem. Meaning instances where some agents already have ownership over some items, think of people trading their cars for example. This will stress what we desire from our allocations since now we have to consider that agents, who are having ownership, might not want to participate if we do not present them with acceptable terms. Furthermore, other models consider agents to be able to take more than one item, presenting with them some upper and lower bounds of consumption. Another modification of the problem would be stating some agents' priority over others, as in the example of the company some members can have priority

over others due to their seniority.

With all this considered, it should not be a surprise to state that the assignment problem is a quite evolving problem with many variations to study. Such as Parameterized Analysis of Assignment Under Multiple Preference by Steindl and Zahavi [6]. In which they explored where there is more than one preference profile, meaning that agents have more than one criteria over items they want. We can imagine such a scenario with houses, where not only the size of the houses but also their location matters. In the following examples, all agents would present their preferences on the location of the house and on the size of the house. It should be easy to imagine other criteria that were taken into account, resulting in more than one preference profile for each agent.

The approach of the thesis was to study assignments, with multiple preferences to be considered. With their fairness of choice being α-*globally-optimal*, which, focuses on Pareto optimality, in a sense that there cannot be agents willing to come to an agreement on trading items that would benefit everyone. However, their model was very restrictive, demanding the resulting assignment to be Pareto optimal in regards to every preference, meaning that if two agents come to each other wanting to trade items because one got pink and the other one green (while they both preferred the other color), such assignment would not be considered Pareto optimal, even if all their other preferences suggested that they would not swap their item with one another. This concept was presented as an assignment that is α-*globally optimal*. Where α specifies the required number of such layers, so that assignment will be α-*globally-optimal*. Pareto optimal layer means that if you look at a specific preference (color, for example), the corresponding layer must be Pareto optimal. This means that two or more agents would not be motivated to swap their items for better results in that quality (swapping pink for a green item, for example). That would be a Pareto optimal layer (in regards to color preference). As we can imagine 3-*globally-optimal* assignment will always be 2-*globally-optimal* as well, but not vice versa.

Here lies the question if it is really necessary for an assignment to be α-*globally-optimal*. Perhaps an assignment could be acceptable if agents are satisfied in the majority of their preferences. After all, if such an instance occurs, agents would still not be willing to trade so we could talk about a different kind of Pareto optimality, but Pareto optimality nonetheless.

Ultimately, one of the key ideas of this thesis is to reduce multiple preference profiles into just one. Some information would be lost during such process. But merging more criteria into just one rating is something reasonable as it is already done in rating, or ultimately in price tagging items.

The core of this thesis is the assignment problem which has been studied in both economical and informatics literature. There is also more than one approach to what we are looking for in an algorithmic finding of assignment, the main ones are time complexity, fairness, and how well is the algorithm resistant to cheating. We will be mainly focusing on time complexity and on fairness. But we will try not to neglect the other parts of the problem in the overall discussion.

Aims

We will start by reflecting on the key concepts used throughout this thesis. Such as assignment problem fairness, and Pareto optimality.

The thesis builds upon the work of Steindl and Zahavi, their goal was to find a solution of the assignment problem under multiple preferences that is Pareto optimal. Coined the term *α-globally-optimal assignment*. That has proven to be (by them) an NP-hard problem, however, the problem limits could be relaxed to find a solution in a more effective way (than NP-hard).

More specifically, this work aims to find an easier to compute algorithm that will in some defined way estimate the *α-globally-optimal assignment*, or find at least some solution if *α-globally-optimal assignment* solution does not exist. Namely, we will have to define different optimality.

In the following work, we will "reduce" assignment problem under multiple preferences to an assignment problem with non-strict profiles, from that point onward we will be analyzing solutions for the Assignment problem with non-strict profiles.

Chapter 1

Preliminaries and Relevant work

Assignment problem and fairness

We should go right ahead by defining the assignment problem, there is no reason why not to use the same definitions as in Parameterized Analysis of Assignment Under Multiple Preferences. [6]. We need to define assignment problem, assignment, and Pareto optimality.

The Assignment problem. An instance of the assignment problem is a triple (A, I, P) where A is a set of n agents $\{a_1, \ldots, a_n\}$, I is a set of m items $\{b_1, \ldots, b_m\}$, and $P = (\prec_{a_1}, \ldots, \prec_{a_n})$, called the *preference profile*, contains the preferences of the agents over the items, where each $\prec_{a_i}$ encodes the preferences of a_i and is a linear order over a *subset* of I (preferences are allowed to be incomplete). We refer to such linear orders as *preference lists*. If $b_j \prec_{a_i} b_r$, we say that agent a_i *prefers* item b_r over item b_j, and we write $b_j \preceq_{a_i} b_r$ if $b_j \prec_{a_i} b_r$ or $b_j = b_r$. Item b is *acceptable* by agent a if b appears in a's preference list. An *assignment* is an allocation of items to agents such that each agent is allocated at most one item, and each item is allocated to at most one agent. Since the preferences of the agents may be incomplete or the number of items may be smaller than the number of agents, some agents may not have available items to be assigned to. In order to deal with this case, we define a special item $b_\emptyset$, seen as the least preferred item of each agent, and it will be used as a sign that an agent is not allocated an item. Throughout this thesis, we assume that $b_\emptyset$ is not part of the input item set, and that it appears at the end of every preference list (we will not write $b_\emptyset$ explicitly in the preference lists). We formally define assignments as follows:

▶ **Definition 1.1.** *Let $A = \{a_1, \ldots, a_n\}$ be a set of n agents and let $I = \{b_1, \ldots, b_m\}$ be a set of m items. A mapping $p : A \to I \cup \{b_\emptyset\}$ is called an* assignment *if for each $i \in [n]$, it satisfies one of the following conditions:*

1. $p(a_i) = b_\emptyset$.
2. *Both $p(a_i) \in I$ and for each $j \in [n] \setminus \{i\}$, $p(a_i) \neq p(a_j)$.*

We refer to p as *legal* if, for each $i \in [n]$, it satisfies $p(a_i) = b_\emptyset$ or $p(a_i) \in I$ is acceptable by a_i. For brevity, we will omit the term "legal", referring to a legal assignment just as an assignment.[1] Moreover, when we write a set in a preference list, we assume that its elements are ordered arbitrarily unless stated otherwise. In the Assignment problem, we are given such triple (A, I, P), and we seek a *Pareto optimal* assignment.

[1] All the "optimal" assignments that we construct in this thesis will be legal in a sufficient number of layers, where they are claimed to be Pareto optimal.

Pareto Optimality. There are different ways to define the optimality of assignments, but the one that received the most attention in the literature is *Pareto optimality*. Informally speaking, an assignment p is *Pareto optimal* if there does not exist another assignment q that is "at least as good" as p for all agents and is "better" for at least one agent. It is formally defined as follows.

▶ **Definition 1.2.** *Let $A = \{a_1, \dots, a_n\}$ be a set of agents and let I be a set of items. An assignment $p : A \to I \cup \{b_\emptyset\}$ is* Pareto optimal *if there does not exist another assignment $q : A \to I \cup \{b_\emptyset\}$ that satisfies:*

1. *$p(a_i) \preceq_{a_i} q(a_i)$ for every $i \in [n]$.*
2. *There exists $i \in [n]$ such that $p(a_i) \prec_{a_i} q(a_i)$.*

If such an assignment q exists, we say that q Pareto dominates *p.*

Now that we have the most important definitions established. We can examine *Pareto optimality*. This has achieved great renown in both game theory [7], but also in economics [8]. However, it is important to note that even though the direction of searching for a *Pareto optimal* solution is the focus of this thesis as well as in the Parameterized Analysis of Assignment Under Multiple Preferences [6], it is not, however, the only relevant one.

Social utility, which is the sum of all agents' utility is another method studied with great attention [9]. But even maximizing all agents' utility might not be desirable in all cases, the motivation here is the fact that a fairer allocation indirectly implies less variation in taxes, which can be desirable in a situation where the (implicit) individual agent budgetary constraints make payment of large taxes unrealistic [10]. This should be easy to imagine in a case where goods are taxed, having now to consider both the upside and downside of allocating the item.

Another problem might arise from measuring social utility for every agent, for example, if agents have incomplete preference profiles then one agent might only accept three items but the other one would accept five items. Is it the same social utility if the first agent receives his second choice and the other one his third choice (middle of their preference profiles), or is it the same social utility if both of them receive their third-best choice? One of the benefits of *Pareto optimality* is that it is strict and easily definable.

Pareto optimality and Trading graph

Another possibility of how to look at *Pareto optimality* is that it makes it impossible for agents to group after the allocation and agree on a trade of their items because every participant would benefit from such trade. It is intuitive that a solution that would allow this is not great, because if two or more agents could trade items for better results for all, why was not that possibility found in the algorithm in the first place? And the same goes for just one agent who finds out that he would rather take the item that was not allocated to anybody instead of his. Again, if there exists an improvement that only benefits agents, there is no reason for it not to be done.

The idea of agents not being able to find some all-satisfactory trading option is the concept for *Trading graph*, which is an oriented graph. In the graph there will be nodes for every agent and every item, there will also be edges to all items from agents are willing to accept in trade and edges, and lastly, there will be edges from items to agents that have the item allocated. Lastly, items are not allocated points to all agents that accept them.

Cycles in such a graph will represent an option for agents to trade items in a way that benefits every participant. So, the problem will be reduced to constructing a graph and checking if there is a cycle. Formally described as follows.

▶ **Definition 1.3.** *An assignment p admits a* trading cycle $(a_{i_0}, b_{j_0}, a_{i_1}, b_{j_1}, \dots, a_{i_{k-1}}, b_{j_{k-1}})$ *if for each $r \in \{0, \dots, k-1\}$, we have that $p(a_{i_r}) = b_{j_r}$ and $b_{j_r} \prec_{a_{i_r}} b_{j_{r+1 \pmod k}}$.*

▶ **Definition 1.4.** *An assignment p admits a* self loop *if there exist an agent a_i and an item b_j such that b_j is not allocated to any agent by p, and $p(a_i) \prec_{a_i} b_j$.*

▶ **Proposition 1.5** (Folklore; see, e.g., Aziz et al [11, 12])**.** *. An assignment p is* Pareto optimal *if and only if it does not admit trading cycles and self loops.*

By this proposition, the problem of checking whether an assignment admits trading cycles or self loops can be reduced to the problem of checking whether the directed graph defined next contains cycles. For an instance (A, I, P) and an assignment p, the corresponding *trading graph* is the directed graph defined as follows. Its vertex set is $A \cup I$, and there are three types of edges:

- For each $a \in A$ such that $p(a) \neq b_\emptyset$, there is a directed edge from $p(a)$ to a. Namely, each allocated item points to its owner.
- For each agent $a \in A$, there is an edge from a to all the items it prefers over its assigned item $p(a)$ (if $p(a) = b_\emptyset$, a points to all its acceptable items).
- Each item without an owner points to all agents that accept it.

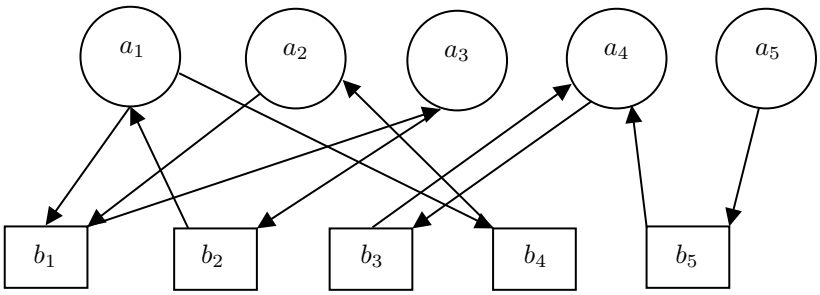
▶ **Proposition 1.6** (Folklore; see, e.g., Aziz et al [11, 12])**.** *. An assignment p is* Pareto optimal *if and only if its corresponding trading graph does not contain cycles.*

By this proposition, given an assignment, we can check if it is Pareto optimal. We can do so by constructing a trading graph and then using DFS [13] to check if the constructed graph does contain cycles. Here we will use the fact that DFS marks vertices as opened and closed. If we from any vertex discover another vertex that is still opened, it must mean that there is a cycle in the graph.

Example. Suppose that $A = \{a_1, a_2, a_3, a_4, a_5\}$ and $I = \{b_1, b_2, b_3, b_4, b_5\}$. Assume that the preferences of the agents are defined as follows.

- $a_1 : b_4 \succ b_1 \succ b_2 \succ b_5$
- $a_2 : b_1 \succ b_4 \succ b_5$
- $a_3 : b_2 \succ b_1$
- $a_4 : b_3 \succ b_5$
- $a_5 : b_5$

Let $p : A \to I \cup \{b_\emptyset\}$ be an assignment such that $p(a_1) = b_2$, $p(a_2) = b_4$, $p(a_3) = b_1$, $p(a_4) = b_5$, and $p(a_5) = b_\emptyset$. The trading graph of the preference profile with respect to p is:



Observe that the agents a_1, a_2 and a_3 admits to the *trading cycle* $(a_1, b_2, a_2, b_4, a_3, b_1)$, and that agent a_4 admits a self loop with b_3. By 1.6, p is not Pareto optimal. If a_1, a_2, and a_3 exchange their items, a_4 gets b_3, and a_5 gets b_5, we have a *Pareto optimal* assignment q in which $q(a_1) = b_4$, $q(a_2) = b_1$, $q(a_3) = b_2$, $q(a_4) = b_3$ and $q(a_5) = b_5$.

Serial dictatorship

A simple assignment mechanism is the greedy *serial dictatorship* mechanism. For a given permutation over the agents, the mechanism takes agents in turns, one in each turn, according to the permutation. That is, the agent which is ordered first allocates its most preferred item, the second allocates its most preferred item among the remaining items, and so on. If at some point, an agent has no available item to allocate in its preference list, it allocates $b_\emptyset$. We say that an assignment p is a *possible outcome* of serial dictatorship if there exists a permutation π such that applying serial dictatorship with respect to π results in p.

▶ **Proposition 1.7** (Abdulkadiroglu and Sönmez [14])**.** *An assignment p is* Pareto optimal *if and only if it is a possible outcome of serial dictatorship.*

▶ **Corollary 1.8.** *A* Pareto optimal *assignment always exists and can be found in polynomial time, and the number of* Pareto optimal *assignments for an instance with n agents is at most $n!$.*

Proposition 1.7yields a surjective mapping from the set of permutations on the agents to the set of *Pareto optimal* assignments. This implies an upper bound of $n!$ on the number of *Pareto optimal* assignments. Observe that this bound is tight: Consider an instance where there is an equal number of agents and items, and all the agents share the same complete preference list. Observe that each permutation gives us a unique assignment after applying serial dictatorship with respect to it. Thus, there exist exactly $n!$ *Pareto optimal* assignments.

Work of Barak Steindl and Meirav Zehavi

As we mentioned previously, our main source is Parameterized Analysis of Assignment Under Multiple Preferences [6].

Generalization of the Assignment Problem. We introduce a generalized version of the Assignment problem where there are ℓ layers of preferences. For each $j \in [\ell]$, we refer to $\succ_{a_i}^{(j)}$ as a_i's preference list in layer j. The *preference profile in layer j* is the collection of all the agents' preference lists in layer j, namely, $P_j = (\succ_{a_1}^{(j)}, \ldots, \succ_{a_n}^{(j)})$. We say that assignment p is Pareto optimal *in layer j* if it is *Pareto optimal* in the single-layered instance (A, I, P_j). To adapt the notion of optimality to the new context, we introduce a natural generalization requiring an assignment to be optimal in a given number of layers.

▶ **Definition 1.9.** *An assignment p is* α-globally optimal *for an instance $(A, I, P_1, \ldots, P_\ell)$ if there exist α layers $i_1, \ldots, i_\alpha \in [\ell]$ such that p is* Pareto optimal *in layer i_j for each $j \in [\alpha]$.*

Thus, the new problem is defined as follows.

α-Globally Optimal Assignment
Input: $(A, I, P_1, \ldots, P_\ell, \alpha)$, where A is a set of n agents, I is a set of m items, P_i is the preference profile in layer i for each $i \in [\ell]$, and $\alpha \in [\ell]$.
Question: Does an α-globally optimal assignment exist?

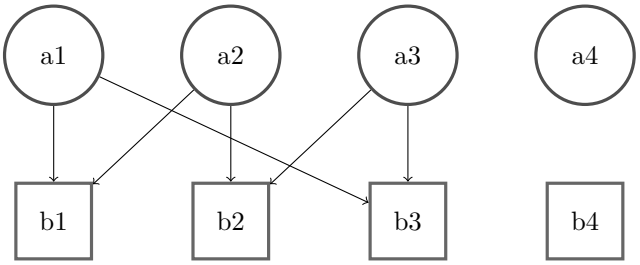


Figure 1.1 The trading graph with respect to p in the fourth layer, where p admits the *trading cycle* $(a_1, b_1, a_3, b_3, a_2, b_2)$.

Consider the following instance, where the agent set is $A = \{a_1, a_2, a_3, a_4\}$, the item set is $I = \{b_1, b_2, b_3, b_4\}$, and there are four layers, defined as follows.

Layer 1:

- a_1 : b_1
- a_2 : $b_3 \succ b_2 \succ b_1$
- a_3 : $b_3 \succ b_1$
- a_4 : $b_2 \succ b_1 \succ b_3$

Layer 2:

- a_1 : $b_2 \succ b_1$
- a_2 : $b_2 \succ b_3$
- a_3 : $b_1 \succ b_2 \succ b_3$
- a_4 : b_3

Layer 3:

- a_1 : $b_2 \succ b_1$
- a_2 : $b_4 \succ b_2 \succ b_1$
- a_3 : $b_1 \succ b_3$
- a_4 : $b_2 \succ b_1 \succ b_3$

Layer 4:

- a_1 : $b_3 \succ b_1 \succ b_2$
- a_2 : $b_1 \succ b_2$
- a_3 : $b_2 \succ b_3$
- a_4 : $\emptyset$

Consider an assignment p in which a_i gets b_i for every $i \in \{1, 2, 3\}$, and a_4 gets $b_\emptyset$. The assignment is 2-globally optimal, since it is *Pareto optimal* in the first two layers. To see this, apply serial dictatorship in the first layer with respect to the permutation (a_1, a_3, a_2, a_4), and in the second layer with respect to the permutation (a_2, a_1, a_3, a_4), and verify that p is the outcome of both runs. In contrast, p is not *Pareto optimal* in the third layer since it admits a self loop among a_2 and b_4 (b_4 is available and is preferred by a_2 over its assigned item b_2). Furthermore, assignment p is not *Pareto optimal* in the fourth layer because it admits a *trading cycle* $(a_1, b_1, a_3, b_3, a_2, b_2)$ (see Figure 1.1): if a_1, a_2 and a_3 trade their items, we get a new assignment q in which a_1 gets b_3, a_2 gets b_1, and a_3 gets b_2; assignment q Pareto dominates p in the fourth layer, and we can verify that it is also *Pareto optimal* in this layer.

Furthermore, the work of Barak Steindl and Meirav Zehavi continues to examine the complexity class and running time of the following problems under different parameters. However, we are trying to take the problem in a different direction.

Game theory and cheating

Now some further things should be mentioned; if we look at the problem from the game theory perspective, we quickly realize that a couple of things should be true for all agents in order for us to be even able to talk about a fair game. We assume that players act rationally and that players play fair, or in this case agents. That means that the agent will not lie about his preferences, since he is rational and obliged to play fair.

Lying about one preference might seem like an irrational thing to do, but if you combine it with prior information about other agents. You could potentially assess your preferences profile accordingly to prior knowledge, to gain an advantage.

For example, let us examine a situation where all agents find item A the most desirable, except for one agent and this agent has the desire and motivation to not be honest, let us call him the villain. Our villain does not find the item A desirable at all. What the villain could do is lie about wanting the item as there would be $(\frac{1}{n})$ chance for him to receive the item A. Now he could trade item A since it is the most desirable item for any other item, thus unfairly increasing his chance of getting the other item *the real item of choice* unfairly, and furthermore, he could trade item A and demand ransom from the agent that would have preferred item A in the first-place.

Another problem is if agents do not act rationally. In such a case, an agent does not act rationally and simply will not state accurately which items are desirable for him. In such cases, we might just allocate items randomly since all of the information we received might not be true or relevant. Of course, it is possible for the information not to be 100% accurate, which is a problem of almost any real life application, but it is beyond the power of an algorithm to fix its input data, so we will treat them as 100% accurate.

Ideally, agents would have no prior knowledge of other agents' preferences, so that it would eliminate options to use that information to their advantage. In some cases, this could be achieved by agents submitting such information Confidentially to the authority. But only in cases where it is reasonable to assume that the preferences of agents could not be effectively predicted. In cases where preferences could be effectively predicted, it becomes a problem, when we take into consideration market research, which is commonly used practice [15], we can see that in many cases, such as when we are distributing items close to a real life marketed goods, in such cases chances of preference prediction among agents would be noticeable. Another case would be if (big) companies were agents, in such cases a lot of information about the company is public (its location, net worth, specialization), so also here it is reasonable to be watching out for such tactics.

In conclusion, a problem in fairness has its origin in the unfairness of players and prior information, which are mostly problems that occur without dependence on the algorithm. It is not that it was impossible to design an algorithm that prevents at least some forms of cheating [16]. But designing an algorithm that would prevent cheating is beyond this thesis's limits, we will, however, discuss possible cheats and exploits of further presented algorithms.

Chapter 2

Finding Algorithm

First, we will define our problem properly, then we will follow up with a transformation of *Assignment problem with multiple layers* to *assignment problem with nonstrict profiles*. Lastly, we will move to find the solution algorithm.

We first present the definition of *Assignment problem with non-strict profiles* so that we know some context for the transformation of the problem.

Definitions

The Assignment problem with non-strict profiles. An instance of the Assignment with equation problem is a triple (A, I, P) where A is a set of n agents $\{a_1, \ldots, a_n\}$, I is a set of m items $\{b_1, \ldots, b_m\}$, and $P = (\prec_{a_1}, \ldots, \prec_{a_n})$, called the *preference profile*, contains the preferences of the agents over the items, where each $\prec_{a_i}$ encodes the preferences of a_i and is a linear order over a *subset* of I (preferences are allowed to be incomplete) But profiles may or may not contains $\sim_{a_i}$ instead of $\prec_{a_i}$. We refer to such linear orders as *preference lists*. If $b_j \prec_{a_i} b_r$, we say that agent a_i *prefers* item b_r over item b_j, and we write $b_j \precsim_{a_i} b_r$ if $b_j \prec_{a_i} b_r$ or $b_j \sim_{a_i} b_r$. And the relations are exclusive, because for any item can be $b_j \prec_{a_i} b_r$ or $b_j \sim_{a_i} b_r$ or $b_j \succ_{a_i} b_r$, but it can be only one of those. Since the preferences of the agents may be incomplete or the number of items may be smaller than the number of agents, some agents may not have available items to be assigned to. To deal with this case, we define a special item $b_\emptyset$, seen as the least preferred item of each agent, and it will be used as a sign that an agent has not allocated any item. Throughout this paper, we assume that $b_\emptyset$ is not part of the input item set, and that it appears at the end of every preference list (we will not write $b_\emptyset$ explicitly in the preference lists). We formally define assignments as follows:

▶ **Definition 2.1.** *Let $A = \{a_1, \ldots, a_n\}$ be a set of n agents and let $I = \{b_1, \ldots, b_m\}$ be a set of m items. A mapping $p : A \to I \cup \{b_\emptyset\}$ is called an* assignment *if for each $i \in [n]$, it satisfies one of the following conditions:*

1. $p(a_i) = b_\emptyset$.
2. *Both $p(a_i) \in I$ and for each $j \in [n] \setminus \{i\}$, $p(a_i) \neq p(a_j)$.*

What the definition says is that the function $p(a_i)$, which is the function representing the item that the agent has assigned to him, is simple, except when $p(a_i) = b_\emptyset$.

Preference tier. Given the triple (A, I, P). If there is $\sim_{a_i}$ relation between any two different items. In a specific profile of an agent. We call them the same *preference tier* in the context of

a given agent. There can be one or more items in one preference tier, if we say one preference tier is higher than another, we mean that all items in such tier are more desirable for the agent than the one from the former preference tier. Similar with the lower preference tier. If some tier is (exactly) one preference tier higher, it means that the preference tier is higher and there are no other preference tiers between the two.

▶ **Definition 2.2.** *Given the agent preference profile and item set (I, P). One preference tier is a set of one or more items where each two different items from the set have one of these conditions met:*

1. *$\sim_{a_i}$ relation between, for some $i \in I$*
2. *In transitive closure of $\sim_{a_i}$ for $i \in I$ relation, there exists $\sim_a$ between the considered items.*

The preference tier is higher (or lower) than the other if between any two items, one from the higher (or lower) tier and one from the other following conditions are met:

1. *$\prec_{a_i}$ (or $\succ_{a_i}$) relation between, for some $i \in I$*
2. *In transitive closure of $\prec_{a_i}$ for $i \in I$ relation, there exists $\prec_a$ (or $\succ_a$) between the considered items.*

We will use the terms preference tier and tier interchangeably.

Pareto Optimality. Informally speaking, an assignment π is *Pareto optimal* if there does not exist another assignment σ that is "at least as good" as π for all the agents, and is "better" for at least one agent. Formally defined, as follows:

▶ **Definition 2.3.** *Let $A = \{a_1, \ldots, a_n\}$ be a set of agents, and let I be a set of items. An assignment $p : A \to I \cup \{b_\emptyset\}$ is* Pareto dominated *if there does not exist another assignment $q : A \to I \cup \{b_\emptyset\}$ that satisfies:*

1. *$p(a_i) \precsim_{a_i} q(a_i)$ for every $i \in [n]$.*
2. *There exists $i \in [n]$ such that $p(a_i) \prec_{a_i} q(a_i)$.*

If such an assignment q exists, we say that q Pareto dominates *p.*

▶ **Definition 2.4.** *If there is no assignment that* Pareto dominates *assignment π, we say that π is Pareto optimal*

Note that the definition of Pareto optimality is almost the same, as for the assignment problem, but now some agents can trade their item for item of the same preference tier. However, trade where all agents just accept a different item of the same preference tier will not result in Pareto dominant assignment.

Trading graph

▶ **Definition 2.5.** *Trading graph $G(\pi)$ is a representation of an assignment π. The construction of such a graph is as follows:*

Nodes:

- *node for every item*
- *node for every agent*

Different edges:

- *black edges: from each item to an agent that the item was assigned*
- *green edges: from each agent to all items that are higher preference tier*
- *blue edges: from each agent to all items that are the same preference tier*

Trading cycle

▶ **Definition 2.6.** *If the trading graph $G(\pi)$ contains a cycle that contains at least one green edge, we refer to such cycle as a* trading cycle.

Self Loop

▶ **Definition 2.7.** *An assignment π admits a self loop if there exist an agent a_i and an item b_j such that b_j is not allocated to any agent by p, and $p(a_i) \prec_{a_i} b_j$.*

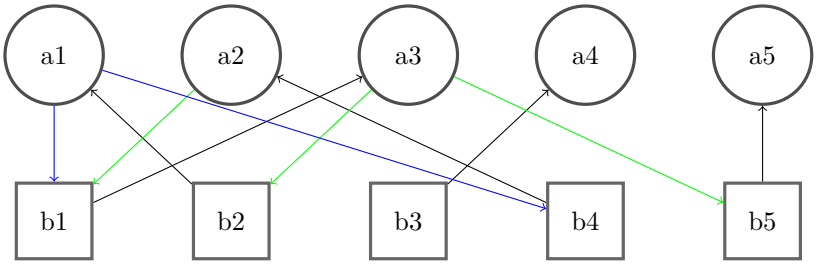
Such occurrence can be also observed as *trading cycle* of length 2.

Example.

agents profiles:

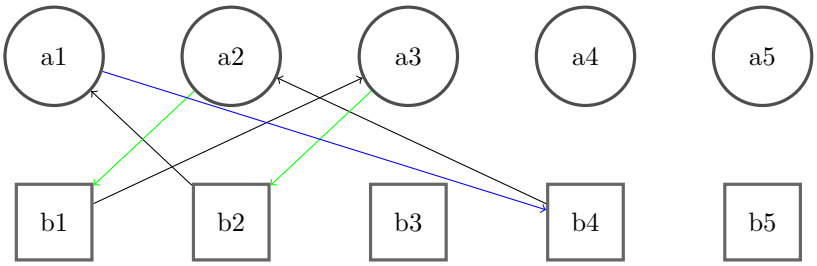
- $a_1 : \ b_4 \sim b_1 \sim b_2 \succ \ ...$
- $a_2 : \ b_1 \succ b_4 \succ \ ...$
- $a_3 : \ b_5 \sim b_2 \succ b_1 \succ \ ...$
- $a_4 : \ b_3 \succ \ ...$
- $a_5 : \ b_5 \succ \ ...$

Let $\pi \colon A \to I \cup \{b_\emptyset\}$ be an assignment such that $\pi(a_1) = b_2$, $\pi(a_2) = b_4$, $\pi(a_3) = b_1$, $\pi(a_4) = b_3$, and $\pi(a_5) = b_5$. The trading graph of the preference profile with respect to π is:

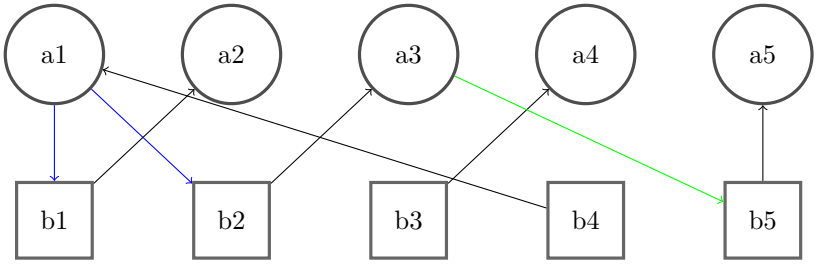


We look for all cycles where at least one edge is green, if there would be all edges blue and black that would not be a *trading cycle* in $G(\pi)$.

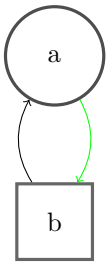
The cycle exists for the following agents and items: $(a_1, b_4, a_2, b_1, a_3, b_2)$.



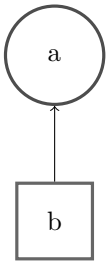
Now that we found a *trading cycle* we can simply reassign the items, in the direction of the trade between agents. Reassignment is done in the direction of blue and green edges so that every agent gets the item that shares a green/blue edge that is part of the *trading cycle*.



In the previous graph, there are no self loops, but if there were some it would be a case of *trading cycle* with length 2.



With graph looking the following after reallocation.



▶ **Proposition 2.8** (Folklore; see, e.g., Aziz et al. [11, 12])**.** *An assignment p is* Pareto optimal *if and only if it does not admit trading cycles and self loops.*

Proof. If there exist a *self loop* or a *trading cycle* in the trading graph $G(\pi)$, then we can rearrange items resulting in the Pareto dominating solution. Therefore assignment π is not Pareto optimal if trading graph $G(\pi)$ contains *self loop* or a *trading cycle*.

If there exists an agent without an item and an item he finds more desirable that situation must result in a self loop. Because it was defined that, the item has a black edge to every agent that accepts it if it is not allocated. The agent has by the definition green edge to all items he finds more desirable than his current choice. So such a situation would result in a cycle consisting of green and black edge which is a *trading cycle* and a *self loop*.

If there exists a cycle of agents and agents such as that at least one agent would get a strictly better item if all agents traded their item for one of the next agents, and none of them would get a less preferred item. Then each agent must have assigned the item he is trading, corresponding black edges. Also, every agent must have an edge to an item he is willing to except in such a

cycle, this is represented in blue and green edges. Lastly, at least one edge must be green, this corresponds to at least one agent getting a more preferred item.

Therefore if there are no *self loops* or a *trading cycles* in $G(\pi)$, then assignment π must be Pareto optimal. ◀

Problem transformation

Relation between Assignment Under Multiple Preference and Assignment problem with non-strict profiles.

Even though we cannot use the methods from the second problem to solve Assignment Under Multiple Preference, we can use these algorithms to approximate the solution. With the approach of synthesizing an instance of Assignment problem with non-strict profiles from an instance of Assignment Under Multiple Preference. Then proceeding to solve the assignment problem with non-strict profiles.

For example, imagine the often used example of students and dormitories [17], [**garlick2018academi**] . Suppose that every student has multiple preferences, one being it's location, size, and number of roommates. following preferences should not be hard to extract from students. Let us say that all of those preferences create a separate preference profile.

If we try to solve this to be Pareto optimal in every layer of preference as Assignment Under Multiple Preference is trying to do, you run into problems such as student A and student B are willing to trade their room due to the more suitable number of roommates (therefore the solution is not Pareto optimal for every layer), but if they did it, it would result in a massive loss for them regarding their desired location.

However, it seems that the solution where the two students could trade because of a small change in the number of roommates for huge dissatisfaction with their desired location seems logical and acceptable.

What we want to express is how much a specific preference is important for a student. In order to do so, we give the agents 10 points that they can distribute to all three things based on how much important they are to them. Not spending all points might or might not be allowed, in this case we allow it. For example $(4, 1, 5)$ $(3, 3, 3)$ or $(8, 1, 1)$ would be all valid distributions of how much a specific attribute is important to them. Let's take an example of preferences being $(5, 4, 1)$ Let order our A, B, C, D items in order of each preference and give the least desired item in that order 0 points and the other +X points where X is how many points agent assigned to this preference. Finally, we can simply add all the points of all items considered, which will result in final ordering.

▶ **Definition 2.9.** *Let $P = \{p_1, \ldots, p_n\}$ be a set of profiles, $V = \{v_1, \ldots, v_n\}$ be a set of vote weights and let I be a set of items. Function value(p_i, b_j), that returns 0 if $b_j \prec b_k$ for each k from $\{1, \ldots, n\} \setminus j$, or Xv_i where is X is the number of k for which $b_j \succ b_k$ is true:*

1. $(b_c) \prec (b_d)$ *if* $\sum_{i=1}^{n} \text{value}(p_i, b_c) < \sum_{i=1}^{n} \text{value}(p_i, b_d)$.
2. $(b_c) \sim (b_d)$ *if* $\sum_{i=1}^{n} \text{value}(p_i, b_c) = \sum_{i=1}^{n} \text{value}(p_i, b_d)$.

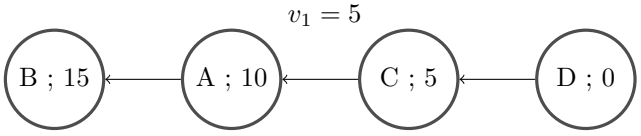
We will refer to this method as point addition.

In the definition, you can see that we can compare any two items to see whether there is a $(\succ, \prec, or \sim)$ relation between them. However, it should not be hard to observe that all $(\succ, \prec, \sim)$ relations are transitive between items. Due to that fact preference profile $P = ([\prec or \sim]_{a_1}, \ldots, [\prec or \sim]_{a_n})$ will give full information about relation between items from the entire set. We use such a preference profile to store the information.

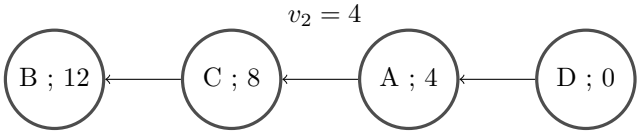
Our situation could look like this:

▶ **Example 2.10.** Given profiles:

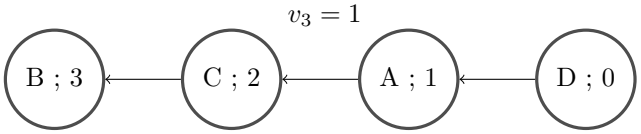
P_1: $B \succ A \succ C \succ D$
$v_1 = 5$



P_2: $B \succ C \succ A \succ D$
$v_2 = 4$

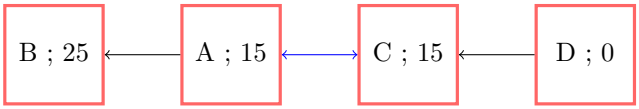


P_3: $B \succ C \succ A \succ D$
$v_3 = 1$



Synthesised profile:

P: $B \succ A \sim C \succ D$



Where the first three graphs represent the three preference profile for each different preference, with corresponding item order and vote weight. Below we have the graph that we get as a result of *point addition*.

As you can see in the final graph, elements A and C have the same amount of points that puts them on the same preference tier, we could force such occurrences not to happen, by simply choosing randomly which item will be more preferred. That would allow us to use the (random) serial dictatorship algorithm to distribute the elements.

However, by removing the possibility of two items being indifferent to one another, meaning equally desirable, in our synthesized profile. The room for different assignments would be lost assignments that could be better, and by better, we mean better in terms of social welfare, in our example we could simply make B $\prec$ C instead of B $\sim$ C, but with B $\sim$ C this agent could trade item B for item C letting someone else take the more desired item. It should be easy to observe that the more items would share the same preference tier the more room for such trades could occur. Accept for the case where all items in agents' preferences profiles are indifferent to one another.

This approach is relatively simple and almost straightforward, definitely, there is a place for more work on this subject. We can come up with different methods on how to transform more preference profiles into just one, perhaps a different problem where multiple preference profiles arise will demand different transformations for a different result. For example, we could even support the equation between elements, trying to find some transformation that creates as many items in the same preference tiers as possible, such changes would leave more variables for different possible solutions of the problem. Or even go as far as to include altruism (one agent would be willing to trade an item if his loss would be small and others' gain will be big).

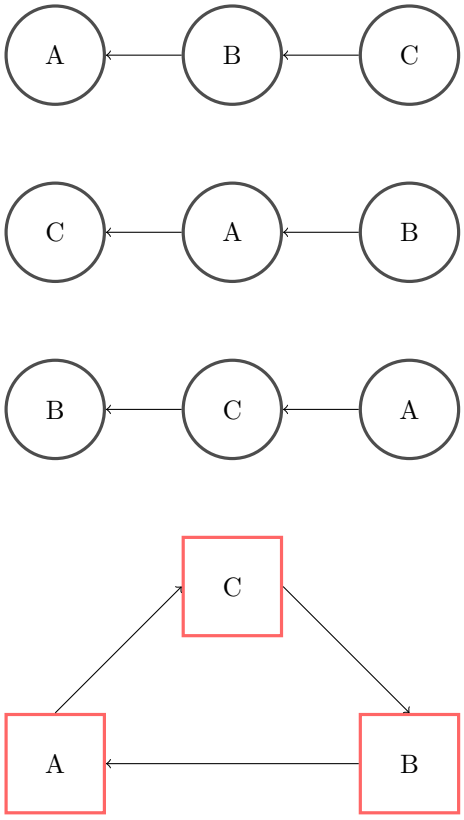
It should be mentioned that for n agents and m items and p preference profiles, we get $O(n \cdot m \cdot p)$ time complexity.

We also tried a simple voting system among profiles. Where one agent prefers item A over B if and only if in all of his preference profile wins vote for item A if compared to item B. If there is no A over B nor B over A preference, then A and B are in the same Preference Tier. Formally defined as follows:

▶ **Definition 2.11.** *Let $P = \{p_1, \ldots, p_n\}$ be a set of profiles, $V = \{v_1, \ldots, v_n\}$ be a set of vote weights, and let I be a set of items. Function vote(p_i, b_j, b_k), that returns 0 if $b_j \prec b_k$ in p_i or v_i if $b_j > b_k$ in p_i. Finally, we construct one preference profile as follows:*

1. $(b_c) < (b_d)$ *if* $\sum_{i=1}^{n} vote(p_i, b_c, b_d) > \sum_{i=1}^{n} vote(p_i, b_d, b_c)$.
2. $(b_c) = (b_d)$ *if* $\sum_{i=1}^{n} vote(p_i, b_c, b_d) = \sum_{i=1}^{n} vote(p_i, b_d, b_c)$.

However, if we try to synthesize preference profiles using this method defined in 2.11 run into the following problem:



(Some profiles can have more than one vote to reflect their importance, but in the following example, every profile has just one vote for simplicity.) The defined relation is not transitive, so we cannot simply order items from the most desirable to the least desirable (even indifference between two items would not have been transitive here, meaning if $A \sim B$ and $B \sim C$ it does not mean that $A \sim C$).

▶ **Lemma 2.12.** *Pareto optimal solution to an assignment problem with non-strict profiles always exists.*

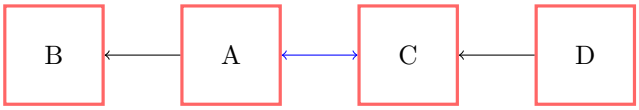
It is not apparent right away that the following problem always must have a solution. For the sake of analysis, it is better to first prove that finding a solution is always possible.

However the idea for proof is quite simple, if there is no Pareto optimal solution, all must have some solution that Pareto dominates the first solution, due to the finite numbers of solutions we must have a cycle of solutions that each Pareto dominates the previous one. Further, we define social welfare score for a solution, that is a number and the number will be strictly bigger when the two solutions compared Pareto dominates one another, due to the fact that the social welfare score number would have to always be bigger we come to the conflict with the premise that there exists an instance of this problem with no Pareto optimal solution.

Let us define the social welfare score for a solution, let social welfare be the sum of welfare for each agent item pair we have. To get a welfare score from an agent, let's arrange his items into tiers where every item in lower tier is less desired and item in the higher tier more desired, it can happen that some items will share the same tier. The score for one agent is simply how many tiers are below his possessed item, so this number will range from 0 to the number of tiers - 1. Formally written as follows:

▶ **Definition 2.13.** *When agent a has assigned item i, his social welfare score is how many preference tiers are below the preference tier that contains item i.*

▶ **Definition 2.14.** *Given the triple of (A, I, P) and the assignment π, with A being agents, I being items, P non-strict preference profile and π being some possible assignment for the (A, I, P). Social welfare $SW(\pi)$ for the assignment π is the sum of all agents' welfare scores, in regard to allocations stated in π.*



For example, for an agent with the following preference profile welfare score for each item would be: B;2, A;1, C;1, D;0.

Proof. Proof of 2.12 Let us take the instance where $n = m$ therefore every agent will have exactly one item allocated and no item will be left. In that case, there is $n!$ solutions, therefore a finite number. For $n < m$ it is $\frac{m!}{(m-n)!}$ and for $n > m$ it is $\frac{n!}{(n-m)!}$, therefore always a finite number.

Due to to fact that every solution is not *Pareto optimal* there must be another solution that Pareto dominates that solution, however, all solutions are not *Pareto optimal* so there must be some other that Pareto dominates the other one and so on. This must lead to a cycle of solutions that each Pareto dominates the previous and is Pareto dominated by the next.

When we compare two solutions if the second one Pareto dominates the first, the second must have a strictly higher social welfare score. This is due to the fact how Pareto domination is defined when solution Pareto dominates it means that some agents will get a more desirable item, but none will take worse item than they currently have, therefore some agents will get an item from a higher tier, but no one will take an item from lower tier, therefore the sum of all agents social welfare will strictly increase.

Now we have a cycle of solutions that all have to have a strictly higher social welfare score than the previous one and strictly lower than the next one. But you cannot have a cycle of numbers that are each strictly bigger than the previous one. For the sake of illustration let $S = \{\pi_1, \ldots, \pi_n\}$ be a set of solutions that form described cycle and and $SW(s_x)$ social welfare score of a specific solution π_i. We see that $SW(\pi_1) < SW(\pi_2), \ldots, SW(\pi_{n-1}) < SW(\pi_n), SW(\pi_n) < SW(\pi_1)$. Therefore $SW(\pi_1) < SW(\pi_n)$ and $SW(\pi_n) < SW(\pi_1)$ which is a contradiction. ◀

Serial dictatorship initial estimation

Basic serial dictatorship solution and why it is not optimal. Serial dictatorship uses some permutation of agents, if not specified it take random permutation, which is this case, that represents the order of agents. The first agent in order is invited to take one item that isn't allocated, and the agent takes his most preferred item that he is able to. Then the same process is repeated in turns for each agent.

Simple serial dictatorship could be used with a simple addition to the algorithm and that is that every time agent has to take an item he either takes his most desired item OR randomly one of the most desired items. Simply named *serial dictatorship approximation*.

Input: Instance Assignment problem with non-strict profiles as (A, I, P), where A is a set of n agents, I is a set of m items, P is the non-strict preference profile.

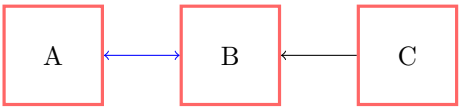
Output: Assignment π.

serial dictatorship estimation

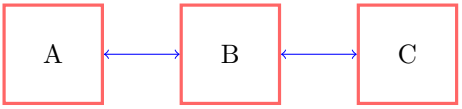
1: apply serial dictatorship on (A, I, P)
2: **return** assignment π

As demonstrated on the following:

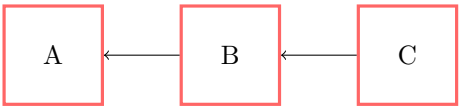
▶ **Example 2.15.** Agent 1:



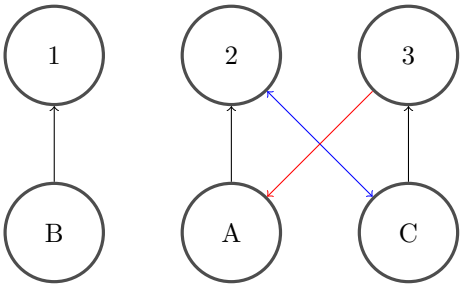
Agent 2:



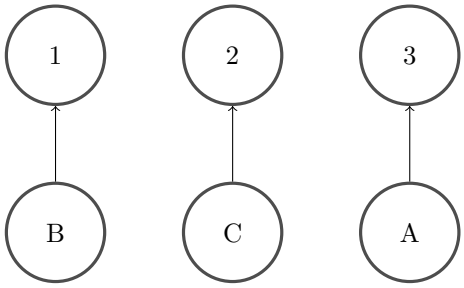
Agent 3:



A Possible result for serial dictatorship in order 1,2,3:



A *Pareto optimal* result:



Even though serial dictatorship is not guaranteed to return *Pareto optimal* result, it can still be used as an estimation. Since its time complexity is for n agents and m items, with time complexity being $O(n \cdot m)$.

The alternative of finding random allocation with time complexity is still $O(n \cdot m)$ due to the fact that preference profiles might be incomplete (some agents might not even accept some items at all), and we still have to track which items are acceptable for each agent. Ultimately results in a similar process algorithm as in serial dictatorship except that agents do not choose their best choice items, they choose a random item that is from their highest tier. Such an option is even less likely to be Pareto optimal and further from the maximum possible social welfare score, where the theoretical maximum possible social welfare would be $n \cdot m$, which is s maximum (even though that theoretical maximum might not exist). Since our goal is to mainly focus on *Pareto optimality* but also to increase social welfare. Which was an important idea in the proof 2.12and will also be one of the driving concepts in the following algorithm.

Another option would be to allocate a null item to every agent (a synthesized null item that is not part of the input and is everyone's least desirable item) that would result in a time complexity of $O(n)$. We receive better time complexity, however, the allocation we would get is the worst possible. Such an allocation would be the farthest possible from the maximum social welfare score.

In conclusion, $O(n \cdot m)$ is not unexpectedly high and still will be overshadowed by the time complexity of algorithms that follows after this procedure.

Algorithmic procedures

In this chapter, we will define procedures that we will use during the following algorithms.

Manifesting a trading cycle. Given a *trading cycle* c in *trading graph* $G(\pi)$ manifesting mean reassign items in π so that such *trading cycle* no longer exists. This is done by moving items in the direction of the *trading cycle*. This can be also viewed as letting agents in mentioned cycle come together and trade items with each other.

▶ **Definition 2.16.** *Given a* trading cycle c *in* trading graph $G(\pi)$ *manifesting this cycle means:*

1. *unallocate all items that have the agents in the trading cycle c has had allocated.*
2. *for each agent that agent allocates the item that he shares a blue or a green edge within trading cycle c.*

▶ **Lemma 2.17.** *By manifesting a trading cycle we increase our social welfare score.*

Proof. Given trading graph $G(\pi_0)$ and trading cycle c, where π_1 is a result from manifesting c. As we can notice when manifesting c, each agent can only get an item he has blue/green edge within c. For an agent to get an item from lower tier that he currently has, he would have to be assigned an item that he has no edge in c, which cannot be the case. Now, there must be at

least one agent that has a green edge to an item in c That agent will get assigned an item from the higher tier of items (since green edges are only to items from the higher tier). Due to this fact, it must be true that $SW(\pi_0)$ ¡ $SW(\pi_1)$ ◀

Finding a trading cycle. Finding a trading cycle in the trading graph $G(\pi)$ literally means to find a cycle. This is achieved by modified DFS that for each node holds open time, close time, and state. If from an opened node we discover another node that is currently opened and not closed, we have found a cycle. If the cycle contains at least one green edge, we have found a trading cycle and stop the procedure.

Finding all trading cycles. Finding a trading cycle in a trading graph is similar to finding a cycle, but we must find all trading cycles, process uses DFS that every time it finds a cycle, it checks if it has at least one edge green, if so it accepts it as a trading cycle and continues to run until it finds all the trading cycle, therefore, until every node is closed.

Update trading cycle c in regards to trading cycle d in trading graph $G(\pi)$. At some point in the execution of the algorithm it can happen that we have more than one trading cycle in the memory. Ideally, we would like to manifest all of them, but it can happen that because we manifest one, we can no longer manifest the other. For that, we use the update procedure, which checks if the updated cycle is still possible to manifest.

▶ **Definition 2.18.** *Given a trading cycles c updating it in regards to trading cycle d in* trading graph $G(\pi)$.

1. *if none of the edges are shared in the two trading cycles ($E(c) \cap E(d) = \emptyset$) end the procedure.*
2. *for any edge that has been removed by manifesting cycle d if any one of these edges has been in c mark c as invalid and end the procedure.*
3. *if some edges used to be green and were changed to blue by manifesting trading cycle d, change those edges also in c.*
4. *check if c has at least one green edge; if not, mark c as invalid.*

All SD cycle update (First algorithm)

The algorithm builds upon the idea of the proof of 2.12. The idea is to find a solution that is not optimal and from that solution find the best one that is Pareto optimal.

Firstly let us use serial dictatorship approximation. As we explored the solution we receive is not guaranteed to be Pareto optimal thus we move to the second part of the algorithm. Where we must find a solution that is Pareto optimal.

For that we will represent our assignment with a graph, on that graph, we will look for trading cycles, we must find all the cycles. We will store information about every *trading cycle* we found and store them in a structure.

In the next step, we will iterate and in each iteration, we take the longest cycle from our structure and manifest it. Then we must update our stored trading cycles. (Keep in mind that every with manifesting the trading cycle yields a different solution, and different assignment.) The updating is done by checking all vertexes if they participate in any other cycle, if so, we must check if their corresponding edges still exist, or still is green. We remove all cycles that are no longer viable (no longer are trading cycles) after our operation. We repeat until the structure containing the trading cycles is empty.

Input: (A, I, P), where A is a set of n agents, I is a set of m items, P is the non-strict preference profile.

Output: *Pareto optimal* assignment π.

Algorithm 1 All SD cycle update

```
1: apply serial dictatorship on (A, I, P)
2: take assignment π
3: compose trading graph G(π)
4: find all trading cycles in G(π) and store them in structure C
5: while C not empty do
6:     manifest biggest trading cycle c
7:     update other cycles in C in regards to c
8:     remove trading cycle c from C
9: end while
10: return Pareto optimal assignment π
```

Proof of correctness. Since we use BFS and serial dictatorship which properties have been already proven, it only remain to prove few claims that we use in the algorithm:

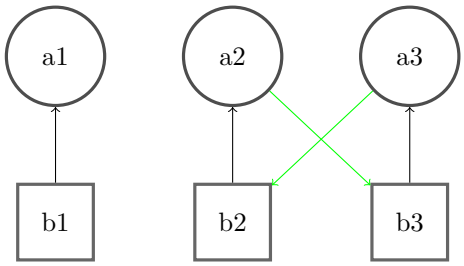
- By removing a biggest cycle we do not create new cycles.
- In updating section (line 7 of the algorithm) we can find and remove all cycles that should perish.
- The algorithm always ends.
- The returned allocation is Pareto optimal.

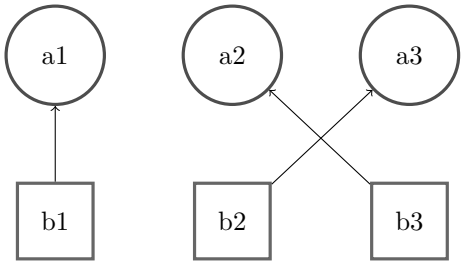
▶ **Lemma 2.19.** *By updating the biggest trading cycle c we do not create new cycles.*

We gather our information from the construction of a trading graph and how the edges will behave during manifesting a trading cycle.

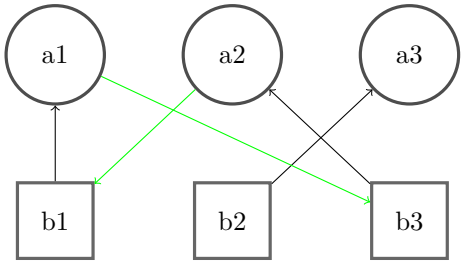
There cannot be a new green edge since items are moved between agents in such a manner that the agent can only get a better item, so every edge to the more desirable item must have already been there, green edges can only disappear. Second, the only way a blue edge can appear is where used to be a blue edge, since every agent will get the same or better preference tier it can happen that the blue edge will transform itself into a green one. But outside this process, new blue edges cannot be created due to the same logic as with green edges. Thirdly, there can and will be created new black edges, but they will not create any cycle, if we removed the longest cycle in the graph.

Proof. Proof of 2.19. For the sake of contradiction let us imagine the following, there exists a longest trading cycle in the trading graph and by its manifestation, we create another cycle by moving the black edge.

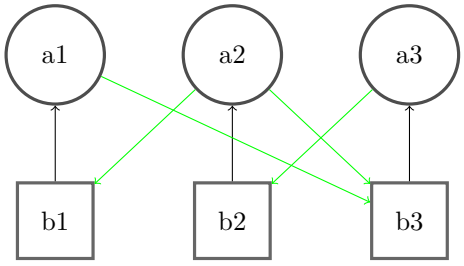




Let us refer to agents and items as inside if they were in the trading cycle c and outside if they were not. Imagine that after the manifestation of the biggest cycle, we got a graph that contains a cycle involving some outside agent(s).



You can see that there must be now two edges (blue or green) that connect the inside agent to the outside item and the outside agent to the inside item. However, since blue or green edges cannot be created by manifesting cycle, those edges must have been there before.



Therefore, those edges helped to create a bigger cycle than the one manifested by combining the cycle we started with and the cycle we found afterward. Therefore, the cycle we processed could not be a biggest, since there is a bigger cycle. ◀

Set of assignments. Another way of representing the problem is a set of assignments. We working in some subset of all the assignments, namely only the ones that are legal, which would be all where all agents have assigned some item from their preference profile or null item. (Preferences profiles might be incomplete, so some agents might not even accept some items.) On such a set, you can observe that manifesting a trading cycle changes the assignment we could imagine the trading cycle as pathways between two assignments. From this observation an alternative route for the proof presented itself:

Proof. Alternative proof of 2.19. This can be observed if we look at the problem as different assignments that include the following agents, where the first trading cycle a changes the assignment from A to B, which created a different trading cycle b which manifesting changes assignment from B to C. We can assume that all of these assignments are legal. Due to the fact that a caused the creation of b it is impossible that there is no overlap between the two cycles. Lets call the cycle cycle created by combination a and b a trading cycle c.

However, manifesting c changes the assignment from A to C. This trading cycle must be the biggest trading cycle of the three. ◀

▶ **Lemma 2.20.** *In updating section (line 7 of the algorithm) we can find and remove all cycles that should perish.*

Proof. Since we have information about all updated agents we can erase all edges that no longer exist, if we delete edge from a cycle the cycle will be removed. If we change all edges in cycle to blue edges the cycle will be removed. ◀

▶ **Lemma 2.21.** *The algorithm always ends.*

Proof. Since there is a finite number of cycles in any graph, there must be a finite number of trading cycles in the trading graph $G(\pi)$. ◀

▶ **Lemma 2.22.** *The returned allocation is Pareto optimal.*

Proof. Thanks to the lemma 2.19 and the fact that at the end of the algorithm structure C must be an empty, trading graph $G(\pi)$ must not contain any trading cycles. Therefore, the returned solution is Pareto optimal. ◀

SD greedy cycle update (second algorithm)

Firstly let us use serial *greedy serial dictatorship algorithm*, we can use a predetermined permutation of agents or use random serial dictatorship, with the exception that every time agent has more items that are his most preferred he will choose randomly one among the best. As we explored the solution we receive is not guaranteed to be Pareto optimal thus we move to the second part of the algorithm. Where we must find a solution that is Pareto optimal.

Afterwords we will construct the corresponding trading graph, in the following steps we will iterate, and in each iteration we will find a trading cycle, manifest such a trading cycle, and update the graph.

We iterate until there are no trading cycles.

Input: Instance Assignment problem with non-strict profiles as (A, I, P), where A is a set of n agents, I is a set of m items, P is the non-strict preference profile.

Output: Pareto optimal assignment π.

Algorithm 2 SD greedy cycle update

```
1: apply serial dictatorship on (A, I, P)
2: while trading cycles in G(π) do
3:     find trading cycle c in G(π)
4:     manifest trading cycle c
5: end while
6: return Pareto optimal assignment π
```

Proof of correctness. We need to prove two smaller claims. Those claims put together are proving the correctness.

- The algorithm will always finish.
- The returned solution is Pareto optimal.

▶ **Lemma 2.23.** *The algorithm will always finish.*

Proof. As highlighted in 2.12, Pareto optimal solution always exists, due to the fact that every time we manifest a trading cycle, this will strictly increase social welfare, which cannot be done infinite amount of times. ◀

▶ **Lemma 2.24.** *The returned solution is Pareto optimal.*

Proof. Since there is condition that there are no trading cycles in trading graph $G(\pi)$ that must be the case. If there are no trading cycles in $G(\pi)$, assignment π must Pareto optimal. ◀

Time complexity

All SD cycle update (first algorithm).

In the first algorithm, we will examine every step, for n agents and m items. For serial dictatorship estimation, we have the time complexity of $O(n \cdot m)$. For computing the trade graph we have a time complexity of $O(n \cdot m)$. However, finding all cycles in an oriented graph is a $O(2^{(n+m)} \cdot (n+m)!)$ time complexity, given that that is the highest number of cycles in a graph. For manifesting trading cycle is $O(n * m)$ cause in worst case all agents and items will change owners and we will have to look in preference profile for each agent and update their edges, also update item edges. Now to find out how many times we will have to iterate we must observe that every trading cycle manifestation must increase the social welfare score by at least one (for at least one agent must get better items). Now as a result of serial dictatorship the first agent will get his best choice item, so he cannot be assigned better item, the second one however could get assigned a better item exactly one tier higher than his. In the "worst case scenario" each agent can have one more tier that he could get items from than the previous. Resulting in $\{0, 1, \ldots, n-1\}$ therefore $O(n^2)$. After each iteration, we will have to check all remaining cycles that share an edge with the just manifested one, there might be more than one check for the same trading cycle as more edges can be shared, resulting in $O(n^2 \cdot (n \cdot m) \cdot 2^{(n+m)} \cdot (n+m)! \cdot (n \cdot m))$. Simplied to $O(n^4 \cdot m^3 \cdot 2^{(n+m)} \cdot (n+m)!)$.

For the time complexity, we get $O(n^4 \cdot m^3 \cdot 2^{(n+m)} \cdot (n+m)!)$.

SD greedy cycle update (second algorithm).

For n agents and m items, the second algorithm has time complexity $O(n \cdot m)$ for serial dictatorship estimation and the time complexity of $O(n \cdot m)$ for trade graph construction. Now for finding a cycle we have the time complexity of $O(n \cdot m)$ using DFS, manifesting a trading cycle is $O(n+m)$ since we might have to update every item and/or every agent. With $O(n^2)$ time complexity (as demonstrated above) possible iterations over trading cycles. Resulting in $O(((n \cdot m) + (n+m)) \cdot n^2)$ time complexity for finding trading cycle, manifesting it, that done $O(n^2)$ times. This simplified is the complexity of $O(m \cdot n^3)$.

For the time complexity, we get $O(m \cdot n^3)$.

Space complexity

All SD cycle update (first algorithm).

For the space complexity, we get $O(2^{(n+m)} \cdot (n+m)! \cdot (n+m))$.

SD greedy cycle update (second algorithm).

For the space complexity, we get $O(n \cdot m)$.

Chapter 3
Discussion

Problem Transformation

So we contemplated use cases where such solutions would be accepted, as we can assume, solutions where two agents would not trade items due to the fact that there are mostly satisfied, but there are still one or few qualities that the other item, is something applicable in reality. As mentioned above such cases is something acceptable for distribution of items, due to the fact that agents would still not be motivated to trade items with each other, which we consider as one of the criteria of optimal assignment.

Serial dictatorship approximation

Both algorithms presented take an existing assignment and then proceed to find the corresponding Pareto optimal one by manifesting trading cycles.

The method how to find the initial assignment was serial dictatorship approximation. Serial dictatorship is an easy way to find Pareto optimal for simple *Assignment Problem* however for *Assignment problem with non-strict profiles* serial dictatorship is not guaranteed to find Pareto optimal Solution as demonstrated in Example 2.15.

First it still allows us to implement some agents' priority over others by letting them choose first in the process. But it is not necessary if the order of agents if not specified, will be at random.

Algorithm

In the finding algorithm part we found out that the SD greedy cycle update (second algorithm) is superior in both time and memory complexity. So from this point onward, we will only discuss the the SD greedy cycle update algorithm and when we mention algorithm we mean the second one presented.

There can be some reduction of the time complexity under the reduction of optimality. For example limiting how many iterations we allow our algorithm to complete. It is important to note that after each iteration we get an assignment as a result, we could return the assignment we have after n iterations. Even though that assignment is not guaranteed to be Pareto optimal.

Strategic behavior

Reasonable solution would be to transfer credibility responsibilities to some other authority or to have some validation steps before the algorithm to make sure the participants have not been trying to exploit the algorithm.

Another way would be disabling trades after the allocation has been made. That could be done by some added mechanism that makes it impossible, or by regulating it by contract for example. However, in some cases, it could simply be impossible by the problem nature. For example, if we were to distribute virtual goods that are linked to a specific account, as most of the in-game purchases are. Namely, anything that has its participants taking form of virtual accounts administrated by authority. In those cases preventing or regulating such trades should not be a problem.

Altruistic behavior

On the other hand, agents might be incentivized to some altruistic behavior, via external or internal motivation. External being for example some reward system for those agents. Internal would be out of the goodness of their heart (in economic theory this can be easily overlooked, but there are people like that as well). In those cases, there are few options to do so.

In case of presented multiple preferences that each agent should rate. (Imagine having 50 points and you have to spend them between the color, length, material, and durability of a ribbon that you would like to wrap your present with.) An agent could rate some of the preferences with 0. What would that achieve is that any time there would be two items with a difference only in the attribute rated with 0, an agent would have no preference over those two items resulting in more room for Pareto optimal assignment which could result in someone else getting their more preferred item. This method would be great if agents had a lot of preference options. It is very likely that given 50 preferences profiles at least one is something some agents do not care about at all. Another motivation for this is that it may be given that agents can only choose a certain amount of preference. An example would look like agents presented with 50 preference option, and incentives to only choose 5 that are the most important to them.

Another method of achieving items more likely to be equal in terms of desirability is to make a threshold for how much an item has to be different in order to be seen as more desirable. for example if in our final synthesized profile items ranges from 0 to 100 points we could say that items that are only 1 point apart from each other are considered equally desirable. Again this might leave more room for potential allocation, ideal example would be some agent getting an item that is 1 less from his choice, but the other agent who will get his item will improve his item score by 100. Now what could happen is that all items would be only 1 point apart from each other, consider the case where the item score ranges from 0 to 100 but there are 500 items. For such cases, we must consider how many items in a row we allow to be considered equal by this process. For example, if we say that by this process there can be only two items made equal in a row it would Mean that from $A \prec B \prec C \prec D$ we can make $A \sim B \prec C \sim D$, or $A \prec B \sim C \prec D$, but not $A \sim B \sim C \sim D$. Another option how to solve this problem would be to limit how many times can this process occur. So that after we make two items equal x times we can no longer make two items equal.

Last and the final option would be to synthesise given preference profiles into one preference profile and ask agents if they would like to do any changes. Agents themselves might reorder the items or mark where they have a preference between the two items, or where they have none. Good part about this agent intervention is that it can be limited in any way and at the same time if the agents do not intervene at all we still have their final preference profile and can run the algorithm.

Possible uses

One of the uses for our algorithm would be the student campus housing problem. Where students would rank each of their preferences here is great that not much information needs to be extracted here as if student specifies a preferred location, the items can just be sorted from the closest to the furthest from their ideal location. Same for number of roommates, how big facility do to which cardinal direction should the windows be. This looks like the ideal scenario for such algorithm because on other terms it would be hard to collect data from every agent on tens perhaps hundreds items considered in each different category. That been said the algorithm is ideal for cases where preferences in one area can be easily sorted and then agent just states his ideal.

Comparison to single preference profile approach.

There might be an argument against using such solutions because this always comes down to one synthesized preference profile, so why not just simply have one preference profile is a question worth posing. The answer is that it will be easier to extract data this way in many cases. We are talking about cases with abundance of items, where agents could be asked to simple order every single one, but it would be much more economical process to only let them rate every criteria they want the item to have. Another reason is that information about every single item might not be available at the time of collecting data from the agents. Take into consideration the scenario where there we want to assign leftover cakes from the bakery at the end of the month. We do not know which cakes will be at our disposal at that time, but we know who like chocolate or vanilla and who is allergic to nuts. Rather then asking agents to sort every single cake from their most to least favorite, it seems a much more acceptable solution to ask them for a couple of preferences and their allergies (in that case you have to make the profile incomplete so that no agent will get a cake he is allergic to).

Another good example will be distributing chores in a company (remember items don't have to have always only positive merits). Where every agent (here an employee) would present their preference in time of the task, nature of the task, if he is willing to work overtime, or languages he can speak which also might be important. Thanks to that information you can easily make an assignment of those tasks. It also highlights the importance of not having to present every item for agents to state their preferences. There might be too many tasks in the company for it to even be comprehensive to put it into a list, also information about some tasks might not be known in advance and they might show up "last minute".

Chapter 4

Conlusion

We started by reintroducing the aspects of the assignment problem. Reflected different approaches to finding assignments and different metrics of fairness. We focused on the *Assignment problem under multiple preference* and acknowledged previous work on the topic. With the intention of finding an assignment, that would satisfy Pareto optimality.

We aim to find the solution to *assignment problem under multiple preference*. We presented transformation of this problem into *assignment problem with non-strict profiles*.

We took a different approach to the problem, instead of finding α-*Globally Optimal Assignment*, which was presented in Parameterized Analysis of Assignment Under Multiple Preference [6]. We "extracted" information from multiple preference profiles into one. During the process, some information was lost. We presented one reduction, *point addition*, but briefly explored other methods and options as well, which could emphasize different needs.

For the solution of *Assignment problem with non-strict profiles*, we presented two algorithms. We present proof of correctness for both algorithms for finding *Pareto optimal* assignment if fully executed. As well as exploring time complexity. SD greedy cycle update achieved time complexity of $O(m \cdot n^3)$ and space complexity of $O(n \cdot m)$. Both of which are superior to other presented algorithm.

For the second algorithm, SD greedy cycle update, we presented another version that reduces time complexity under the cost of optimality, such an algorithm is not guaranteed to return *Pareto optimal* assignment. These measures might be useful if lack of computing power is a problem, or where is not any processing power to spare.

We have discussed the usefulness of the algorithm. The main reasons is, it's versatility, in comparison to only single preference approach, you can easily get more information from agents about their preferences, when they can state multiple of them. This allows agents answers to be much more telling that one preference criteria only. Also in many cases it will be much easier to ask agents in each of their preference their ideal and how much they value that preference. This can be utilized in distributing chores in a company, here every agent would state what time preferences he have, what are his preferred tasks and how much the salary is or isn't important to him. Thanks to that information we can then distribute the chores optimally. Another reason also is that this is a good method of extracting agents' preferences without the need of presenting every single item in advance. This might be needed as in the assigning tasks in company example it is clear that some tasks might not be known during the information gathering step, but they present themselves "last minute".

For that, there are measures that could be done before the assigning (making sure that agents know as least information about other agents as possible), or after the assigning (forbid trading items after assignment).

Future work.

Firstly, we started with synthesizing preference profiles into one, there are other methods, which might be used and when facing a specific problem where the presented results are not sufficient, or optimal, that would be the first place to look for improvements.

Lastly, the assignment problem as a social choice problem is not only algorithmic problem. As new knowledge might be discovered in game theory, economy, or other areas there might be some hard to forsee motivation for having to implement new ideas.

Bibliography

1. HOSSEINI, Hadi; LARSON, Kate; COHEN, Robin. *Random Serial Dictatorship versus Probabilistic Serial Rule: A Tale of Two Random Mechanisms*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1503.01488.
2. BOUVERET, Sylvain; CHEVALEYRE, Yann; MAUDET, Nicolas. Fair Allocation of Indivisible Goods. In: BRANDT, Felix; CONITZER, Vincent; ENDRISS, Ulle; LANG, Jérôme; PROCACCIA, Ariel D. (eds.). *Handbook of Computational Social Choice*. Cambridge University Press, 2016, pp. 284–310. Available from DOI: 10.1017/CBO9781107446984.013.
3. PROCACCIA, Ariel D. Cake Cutting Algorithms. In: *Handbook of Computational Social Choice, chapter 13*. Cambridge University Press, 2015.
4. THAKRAL, Neil. *The public-housing allocation problem*. 2016. Tech. rep. Technical report, Harvard University.
5. KLAUS, Bettina; MANLOVE, David F.; ROSSI, Francesca. Matching under Preferences. In: BRANDT, Felix; CONITZER, Vincent; ENDRISS, Ulle; LANG, Jérôme; PROCACCIA, Ariel D. (eds.). *Handbook of Computational Social Choice*. Cambridge University Press, 2016, pp. 333–355. Available from DOI: 10.1017/CBO9781107446984.015.
6. STEINDL, Barak; ZEHAVI, Meirav. Parameterized Analysis of Assignment Under Multiple Preferences. In: ROSENFELD, Ariel; TALMON, Nimrod (eds.). *Multi-Agent Systems - 18th European Conference, EUMAS 2021, Virtual Event, June 28-29, 2021, Revised Selected Papers*. Springer, 2021, vol. 12802, pp. 160–177. Lecture Notes in Computer Science. Available from DOI: 10.1007/978-3-030-82254-5_10.
7. LUC, Dinh The. Pareto Optimality. In: *Pareto Optimality, Game Theory And Equilibria*. Ed. by CHINCHULUUN, Altannar; PARDALOS, Panos M.; MIGDALAS, Athanasios; PITSOULIS, Leonidas. New York, NY: Springer New York, 2008, pp. 481–515. ISBN 978-0-387-77247-9. Available from DOI: 10.1007/978-0-387-77247-9_18.
8. STIGLITZ, JOSEPH E. Pareto Optimality and Competition. *The Journal of Finance*. 1981, vol. 36, no. 2, pp. 235–251. Available from DOI: https://doi.org/10.1111/j.1540-6261.1981.tb00437.x.
9. SINHA, Abhinav; ANASTASOPOULOS, Achilleas. A General Mechanism Design Methodology for Social Utility Maximization with Linear Constraints. *SIGMETRICS Perform. Eval. Rev.* 2014, vol. 42, no. 3, pp. 12–15. ISSN 0163-5999. Available from DOI: 10.1145/2695533.2695538.

10. COLE, Richard; GKATZELIS, Vasilis; GOEL, Gagan. Mechanism Design for Fair Division: Allocating Divisible Items without Payments. In: *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 2013, pp. 251–268. EC '13. ISBN 9781450319621. Available from DOI: 10.1145/2492002.2482582.

11. AZIZ, Haris; HAAN, Ronald de; RASTEGARI, Baharak. Pareto Optimal Allocation under Uncertain Preferences. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 77–83. Available from DOI: 10.24963/ijcai.2017/12.

12. AZIZ, Haris; BIRO, Peter; HAAN, Ronald de; RASTEGARI, Baharak. Pareto optimal allocation under compact uncertain preferences. In: *Thirty Third AAAI Conference on Artificial Intelligence (01/02/19)*. 2018. Available also from: https://eprints.soton.ac.uk/425734/.

13. TARJAN, Robert. Depth-first search and linear graph algorithms. *SIAM journal on computing*. 1972, vol. 1, no. 2, pp. 146–160.

14. ABDULKADIROGLU, Atila; SONMEZ, Tayfun. Random Serial Dictatorship and the Core from Random Endowments in House Allocation Problems. *Econometrica*. 1998, vol. 66, no. 3, pp. 689–702. Available also from: https://ideas.repec.org/a/ecm/emetrp/v66y1998i3p689-702.html.

15. MCGIVERN, Yvonne. *The practice of market research: an introduction*. Pearson Education, 2009.

16. HU, Haiyang; LI, Zhongjin; HU, Hua. An anti-cheating bidding approach for resource allocation in cloud computing environments. *J Comput Inf Syst*. 2012, vol. 8, no. 4, pp. 1641–1654.

17. PERACH, Nitsan; POLAK, Julia; ROTHBLUM, Uriel G. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the technion. *International Journal of Game Theory*. 2008, vol. 36, no. 3, pp. 519–535.