NTNU

Department of Computer Science

TDT4501 — Specialization project

Matroids and fair allocation

Author:
Andreas Aaberge Eide

Supervisor:
Magnus Lie Hetland

December 16, 2022

Chapter 1

Introduction

Fair allocation is the problem of assigning a set of items, either goods or chores, to a set of agents such that each agent is happy with what they obtain. This is an age-old problem in economics, especially for divisible items. In the divisible case, known as cake cutting, fair allocations always exist and can be computed efficiently. Fair allocation with indivisible items is a tougher nut to crack, but computer science offers a new way to approach it with algorithmic fair allocation.

Matroids are mathematical structures that are important in combinatorial optimization, and are known in computer science for being a generalization of problems for which the greedy algorithm works. In the context of fair allocation, matroids can be used to model the valuation functions of agents. When the valuation functions are matroid rank functions, efficient algorithms have been found that deliver well on fairness, efficiency and truthfulness. They can also be used to model item constraints.

This report presents an overview of fair allocation with an emphasis on the ways matroids have been applied to the field. The many different ways of defining a matroid are also discussed. Part 3 presents Knuth's classic algorithm for generating (random) matroids, and the results of running this algorithm with different input is also shown. There does not currently exist any (?) good tools for generating and using random matroids — one goal of this project was to develop a Julia library for this purpose. A Julia implementation of the algorithms discussed can be found in the appendix. Finally, a few fair allocation algorithms for problem instances involving matroids are shortly presented and their demands of such a library discussed.

Chapter 2

Background

2.1 Fair allocation

Fair allocation is the problem of assigning a set of items, either goods or chores, to a set of agents such that each agent is happy with what they obtain. This is an age-old problem in economics, especially for divisible items. In the divisible case, known as cake cutting, fair allocations always exist and can be computed efficiently. Fair allocation with indivisible items is a tougher nut to crack, but computer science offers a new way to approach it with algorithmic fair allocation.

A fair allocation instance consists of m indivisible items $\mathcal{M} = \{1, \ldots, m\}$ and n agents $\mathcal{N} = \{1, \ldots, n\}$. An allocation is an n-partition of $\mathcal{M}$, $\mathbf{X} = (X_1, \ldots, X_n)$, where each $X_i \subseteq \mathcal{M}$ is the bundle of items allocated to agent i. There is no sharing allowed; if $i \neq j$, $X_i \cap X_j = \emptyset$, meaning each item is allocated to at most one agent. If $\bigcup_{i \in \mathcal{N}} X_i \neq \mathcal{M}$, that is, not all items are allocated, then $\mathbf{X}$ is a partial allocation.

Each agent i has a valuation function $v_i : 2^M \to \mathbb{R}$, where 2^M denotes the Power set of M, that assigns a value to each possible bundle of items. The *valuation profile* is the vector of valuation functions $\mathbf{v} = (v_1, \ldots, v_n)$. The marginal gain agent i enjoys by adding good g to their bundle X_i is given by $\Delta_i(X_i, g) = v_i(X_i \cup \{g\})$.

In the simple case, the valuations are additive, meaning

$$v_i(S) = \sum_{e \in S} v_i(\{e\}).$$

The valuation functions are typically also assumed to be monotonic – adding an

extra good to an agent's allocation will never decrease the bundle value. More general valuation functions exist. Of special interest in this report are binary submodular valuation functions, which are equivalent to matroid rank functions (MRFs). If the valuation function v_i is an MRF, it has the following properties:

(MRF1) $f_i(\emptyset) = 0$.

(MRF2) For all agents $i \in \mathcal{N}$ and all goods $g \in \mathcal{M}$, $\Delta_i(X_i, g) = v_i(X_i \cup \{g\}) - v_i(X_i) \in \{0, 1\}$. That is to say, the marginal value gain of adding another element to a bundle is either 1 or 0.

(MRF3) For all agents $i \in \mathcal{N}$, all bundles X, Y such that $X \subseteq Y \subseteq \mathcal{M}$, and all goods $e \in E \setminus Y$, we have $\Delta_i(X, e) \geq \Delta_i(Y, e)$. In other words, adding a good to a smaller bundle will never result in a lower marginal gain than adding it to a larger bundle.

When evaluating an allocation, there are three aspects to consider: *Fairness*, *efficiency* and *truthfulness*. Fairness can be evaluated in terms of envy-freeness or proportionality. An allocation $\mathbf{X}$ is envy-free if, for any two agents $i, j \in N$, we have that $v_i(X_i) \geq v_i(X_j)$. Checking if a given instance admits an EF allocation is an NP-complete problem, even for $\{0, 1\}$- or $\{0, -1\}$-valued instances, and there is often no feasible EF allocation. There is no way, for instance, to allocate one item between two agents in an envy-free manner. Therefore, the research is typically focused on relaxations of EF, the two most prominent being *envy-freeness up to one good* (EF1) and *envy-freeness up to any good* (EFX).

An allocation $\mathbf{A}$ is proportional, on the other hand, if $v_i(A_i) \geq v_i(M)/n$. That is to say, each agent receives at least their proportional share of the total value. This does not take into account what other agents receive, as opposed to EF. If an allocation is envy-free, then it is also proportional.

An allocation where no agent receives any good is trivially envy-free, but is very inefficient. Therefore, allocation algorithms typically make some guarantees with regards to efficiency as well. Allocation efficiency is concerned with how the agent happiness has been maximized. This can be measured in various ways:

The utilitarian welfare of an allocation $\mathbf{X}$ is the total happiness of all agents, given by

$$\sum_{i \in N} v_i(X_i).$$

By maximizing this you maximize the total happiness, hence the term utilitarian.

The egalitarian welfare of an allocation $\mathbf{X}$ is the happiness of the least happy agent:

$$\min_{i \in N} v_i(X_i).$$

Maximize this and you maximize the lowest happiness.

Nash welfare is a compromise between egalitarian and utilitarian welfare, given by

$$\prod_{i \in N} v_i(X_i).$$

An allocation that maximizes Nash welfare is said to be MNW. Such an allocation exhibits very nice fairness properties.

The final property is truthfulness, and is a game-theoretic concern that determines the degree to which agents can benefit from lying about their true preferences over the items being allocated. In a truthful system, the best course of action for each agent is to be perfectly honest. Such a system is considered *strategyproof*.

An allocation rule (for instance, MNW) is group strategyproof (GSP) if there does not exist valuation profiles $\mathbf{v}, \mathbf{v}'$ and a group of agents $\mathcal{C} \subseteq \mathcal{N}$ such that

$$\begin{aligned} v'_k &= v_k & \forall k \in \mathcal{N} \setminus \mathcal{C}, \\ v_j(A'_j) &> v_j(A_j) & \forall j \in \mathcal{C}, \end{aligned}$$

where $A' = f(\mathbf{v}'), A = f(\mathbf{v})$. GSP is a stronger criterion than strategyproofness.

We have hitherto described fair allocation in an unconstrained setting. However, for many practical use cases, there may be good reasons to place constraints on which bundles are legal. For instance, connectivity constraints, in which only bundles of items which are connected according to a supplied connectivity graph is legal. Conflict constraints are another example, in which the item graph describes which items can never co-exist in the same bundle. Of special interest in this (I still don't know what to call this piece of text), however, are matroid constraints, in which we require each allocated bundle to be an independent set of some common matroid $\mathfrak{M}$. Biswas and Barman [1] showed that EF1 allocations exist and can be found efficiently when agents have identical valuations and $\mathfrak{M}$ is laminar, but it is still an open question what fairness guarantees can be made for instances with more general constraint matroids or valuation functions.

2.2 Matroids

A cryptomorphism is a mathematical structure that can be defined or axiomatized in several different ways, whose equivalence is not obviously apparent. A matroid is the prime example of a cryptomorphism, as they can be defined in a number of ways. Which definition is most applicable will depend on the use case. In this section, I will describe some of the ways to define a matroid.

A matroid in terms of its independent sets. Perhaps the most common and intuitive way to characterize a matroid is as an *independence system*. An indepence system is a pair $(E, \mathcal{S})$, where E is the *ground set* and S is the set of *independent sets*, such that

1. E is a set of elements, $E \neq \emptyset$, and
2. $\mathcal{S}$ is a subset of $\mathcal{P}(E)$.

A matroid $\mathfrak{M} = (E, \mathcal{S})$ is an independence system that satisfies the following three additional properties:

(I1) $\emptyset \in \mathcal{S}$.

(I2) If $A \subseteq B$ and $B \in S$, then $A \in \mathcal{S}$. This property is called being closed under inclusion.

(I3) If $A, B \in \mathcal{S}$ and $|A| > |B|$, then there exists an $a \in A$ such that $B \cup \{a\} \in \mathcal{S}$.

Sets that are not independent are *dependent sets*. A useful direct result of (I3) is that all subsets of an independent set are themselves independent. This is known as the **hereditary property**.

A matroid in terms of its bases. A *basis* of a matroid $\mathfrak{M} = (E, S)$ is a maximal independent set for $\mathfrak{M}$, that is, a set in S that cannot have any element from E added to it without becoming dependent. We can define a matroid as $\mathfrak{M} = (E, \mathcal{B})$, where $\mathcal{B}$ is the set of bases for $\mathfrak{M}$. It is easy to see that this is equivalent to the definition of the matroid in terms of its independent sets, as we can use the hereditary property to produce the set of independent sets from the set of bases.

The bases of a matroid have some interesting properties:

(B1) All bases have the same size.

(B2) If $B_1, B_2 \in \mathcal{B}$, then for every $b_1 \in B_1 \setminus B_2$ there exists a $b_2 \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x_1\}) \cup \{x_2\}$ is a basis. This is known as the **basis exchange property**.

A matroid in terms of its rank function. Given a matroid $\mathfrak{M}$ on a set of elements E, the *matroid rank function* $r : 2^E \to \mathbb{Z}^*$ accepts a set $S \subseteq E$, and gives the size of the largest subset of S that is an independent set in $\mathfrak{M}$. It follows that $r(S) \in \{0, \ldots, |B|\}, B \in \mathcal{B}$, for all $S \subseteq E$. $|B|$ is said to be the rank of the matroid itself, $r(\mathfrak{M})$. The properties of the MRF r is described above. By enumerating all subsets of E whose rank equals its size, we find the independent sets of $\mathfrak{M}$. Hence, $\mathfrak{M} = (E, r)$ is equivalent to the independent sets definition of a matroid.

A matroid in terms of its circuits. The *nullity* of a set $S \subseteq E$, given by $n(S) = r(S) - |S|$. It is obvious that $0 \leq n(S) \leq |S|$. A *circuit* is a dependent set S with $n(S) = 1$, that is, an independent set with one "redundant" element added to it. 1

A matroid in terms of its closed sets. Knuth [2] [2] uses this definition of a matroid in his algorithm for generating random matroids, which is described in part 3.

A matroid $\mathfrak{M} = (E, \mathcal{F})$ is a ground set of elements E along with a family $\mathcal{F}$ of subsets of E, satisfying the following axioms:

(C1) $E \in \mathcal{F}$.

(C2) If $A, B \in \mathcal{F}$, then $A \cap B \in \mathcal{F}$.

(C3) If $A \in \mathcal{F}$ and $a, b \in E \setminus A$, then b is a member of all sets in $\mathcal{F}$ containing $A \cup \{a\}$ if and only if a is a member of all sets in $\mathcal{F}$ containing $A \cup \{b\}$.

$\mathcal{F}$ is the set of all *closed sets* of $\mathfrak{M}$. A closed set is a set whose cardinality maximal for its rank, meaning you cannot add any element to it without in-

creasing its rank. Some other useful properties of closed sets that will be useful in our implementation of Knuth's algorithm later:

(C4) If $S \in \mathcal{F}$ is a closed set for $\mathfrak{M}$, and $r(S) = |S|$ (or, equivalently, $n(S) = 0$), then S is also an independent set for $\mathfrak{M}$.

(C5) E is a closed set of $\mathfrak{M}$, such that $r(E) = r(\mathfrak{M})$. E is the closure (DEFINE) of all bases in $\mathcal{B}$.

Property (C4) is obvious, as if there was a way to increase the cardinality of S without increasing the rank, S would not be closed, and the cardinality of a set can not be lower than its rank (this follows from the fact that r is an MRF). By definition, if the rank of a set equals its cardinality, then it is an independent set. An independent set is also a closed set if adding any element to it produces another, larger independent set.

Types of matroids. 2

A very simple type of matroid that will show up in the analysis is the *uniform matroid*. The uniform matroid of rank $r \geq 0$ over a ground set E of size $|E| = n \geq r$, denoted by $U_{r,n}$, is a matroid whose independent sets are all sets $S \subseteq E$ such that $|S| \leq r$. In other words, every subset of E of size up to r is an independent set. If every subset $S \subseteq E$ is independent, which is equivalent to $r = n$, we write U_n.

Chapter 3

Random matroids

In his 1974 paper, Knuth [2] describes a mechanism for generating random matroids in terms of their closed sets. The definition of a matroid in terms of its closed sets can be found in part 2.2.

Let $\mathfrak{M} = (E, \mathcal{F})$ be a matroid, where E is the ground set and $\mathcal{F}$ the family of closed sets of $\mathfrak{M}$. Since $\mathcal{F}$ is a subset of the 2^E, the number of possible such families (and thus different matroids over E) is bounded by $2^{2^{|E|}}$. Therefore it is infeasible to look at all possible families of closed subsets over E. Knuth uses a bottom-up approach instead, constructing $\mathcal{F}$ by starting with the smallest closed sets and incrementally constructing the larger ones. This mechanism is presented in Algorithm 1.

Knuth-Matroid accepts two arguments, the ground set of elements for the matroid, and a sequence X of "enlargements," where X_{r+1} is the set of enlargements two apply at step $r + 1$. The algorithm relies on two subroutines, Enlarge and Superpose, which will be explained shortly. The algorithm incrementally constructs a list of sets of subsets of E, each element a family $F_i \subseteq 2^E$ of closed sets of rank i. When the algorithm terminates, the list $\mathrm{F} = [F_0, \ldots, F_r]$ is returned, where F_i is the family of closed sets of rank i. In the paper, Knuth shows that $\bigcup_{i=0}^{r} F_i = \mathcal{F}$, and so the algorithm outputs valid matroids.

The algorithm initializes with $r = 0$. The only closed set of rank 0 is the empty set, so $F_r = F_0 = \{\emptyset\}$. Inside the loop, on line 3, we produce F_{r+1} from F_r by generating all "covers" of the sets in F_r, meaning all sets in F_r with one more element added from E. Had we only done this for $|E|$ iterations, we would have ended up with the uniform matroid. Not terribly interesting!

Algorithm 1 KNUTH-MATROID(E, X)

Input: The ground set of elements E, and a list of enlargements X.

Output: The list of closed sets of the resulting matroid grouped by rank, $\mathrm{F} = [F_0, \ldots, F_r]$, where F_i is the set of closed sets of rank i.

```
1  r = 0, F = [{∅}, {∅}]
2  while true
3      F[r + 1] = {A ∪ {a} : A ∈ F[r], a ∈ E \ A}
4      ENLARGE(F[r + 1], X[r + 1])
5      SUPERPOSE(F[r + 1], F[r])
6      if E ∉ F[r + 1]
7          r ← r + 1
8      else
9          return F
```

The ENLARGE subroutine is what allows us to generate arbitrary matroids, by selectively enlarging the family of closed sets of a given rank. The subroutine enlarges F_{r+1} by simply adding the sets supplied to the algorithm in X_{r+1}.

The second subroutine, SUPERPOSE, does the following: If F_{r+1} contains two sets A, B whose intersection $A \cap B \not\subseteq C$, for some $C \in F_r$, replace A, B with $A \cup B$. Repeat until no two sets exist in F_{r+1} whose intersection is not contained within some set $C \in F_r$. Knuth shows that massaging each F_i in this manner gives a valid matroid $\mathfrak{M} = (E, \mathcal{F} = \bigcup_{i=0}^{r} F_i)$, satisfying the properties (C1)-(C3).

```
SUPERPOSE(F_{r+1}, F_r)
 1  for A ∈ F_{r+1}
 2      for B ∈ F_{r+1}
 3          flag ← true
 4          for C ∈ F_r
 5              if A ∩ B ⊆ C
 6                  flag ← false
 7
 8          if flag = true
 9              F_{r+1} ← F_{r+1} \ {A, B}
10              F_{r+1} ← F_{r+1} ∪ {A ∪ B}
11
```

At the end of each iteration, r is incremented by one, unless E is found to be a closed set in F_{r+1}, at which point the algorithm terminates. As described above, E is the closure of the bases of the matroid, so upon reaching this point there is no point in going further: $r + 1$ is the rank of the matroid.

My Julia implementation of KNUTH-MATROID can be found in THE APPENDIX. The algorithm as described produces a matroid $\mathfrak{M} = (E, \mathcal{F})$ expressed in terms of its closed sets, however, for it to be applicable to the study of fair allocation, we need several other properties of the matroid.

Finding the other properties of $\mathfrak{M}$

Given a KNUTH-MATROID$(E, \mathrm{X}) = \mathfrak{M} = (E, \mathrm{F})$, where $\mathrm{F} = [F_0, F_1, \ldots, F_r]$ such that $F_0 \cup F_1 \cup \ldots \cup F_r = \mathcal{F}$, the family of closed sets of $\mathfrak{M}$, and r is the rank of the matroid, we want to find

1. the bases $\mathcal{B}$ of $\mathfrak{M}$,
2. the independent sets $\mathcal{I}$ of $\mathfrak{M}$,
3. the closure function $\mathrm{cl} : 2^E \to \mathcal{F}$,
4. the rank function $\mathrm{rank} : 2^E \to \mathbb{Z}^*$ of $\mathfrak{M}$, and
5. the circuits $\mathcal{C}$ of $\mathfrak{M}$.

Given all these (and a usable API to interface with), we will have arrived at a potentially useful library for interacting with matroids in the context of fair allocation.

We begin with the rank function. Since we have access to the closed sets of $\mathfrak{M}$ grouped by rank, $(\mathrm{F} = [F_0, \ldots, F_r])$, we can check the rank of an arbitrary set $S \subseteq E$ by finding the lowest j such that there exists a $B \in F_j$ where $S \subseteq B$.

```
RANK(S, 𝔐 = (E, F))
1  for i in 0..F.length do
2      for B ∈ F[i] do
3          if S ⊆ B do
4              return i (or F[i] to get cl(S))
```

We can get implement the closure operator for free by simply returning the found set $F[i]$ instead.

Now, let's consider how we can express our matroid as $\mathfrak{M} = (E, \mathcal{B})$, where $\mathcal{B}$ is the set of bases (maximal independent sets) of $\mathfrak{M}$. Recall that a closed set is maximal for its rank, and an independent set is minimal for its rank. (C5) tells us that E is the closure of all bases in $\mathfrak{M}$. We know that (B1) all bases have the same size, and since a basis is an independent set, we know that the rank of a basis equals its size. The rank of the matroid is the rank of a basis in the matroid. Hence, $\mathcal{B} \subseteq \{A \subseteq E : |A| = r\}$, that is, the set of all r-sized subsets of E contains $\mathcal{B}$. In the case of the uniform matroid, this would be an equality and we would be done. But in the general case, since Knuth's algorithm might have enlarged the closed sets at some ranks, certain r-sized subsets of E are in fact dependent sets of a lower rank. To find the bases, we need to find the r-sized subsets of E whose rank is r.

BASES($\mathfrak{M} = (E, \mathrm{F})$)
1 **r** $\leftarrow \mathrm{F}.length - 1$
2 **return** $\{B \subseteq E : |B| = \textsc{Rank}(B, \mathfrak{M}) = r\}$

Results

Knuth generates random matroids by applying random "coarsening" in the ENLARGE step. The randomized version, RANDOM-KNUTH-MATROID, works by applying SUPERPOSE immediately after generating covers, then choosing a random member A of $\mathcal{F}_{r+1}$ and a random element $a \in E \setminus A$, replacing A with $A \cup \{a\}$ and finally reapplying SUPERPOSE. Table 3.1 below shows the average values obtained for various settings of the parameters. The sequence $\mathrm{P} = (p_1, p_2, \ldots)$ holds the number of such coarsening steps to be applied at each iteration of the algorithm (p_0 is always 0 and is omitted).

1000 experiments were conducted for each setting of the parameters. Knuth performed the same experiments with at most thirty trials each, using his ALGOL W implementation on the hardware of the day. The results in this report correspond fairly well with Knuth's results. The resulting rank of the matroid for a given input correlates well for most of the coarsening settings. For instance, with $\mathrm{P} = (5, 2, 0)$, Knuth's implementation produced matroids of rank 3 eight out of ten times, which corresponds well with the roughly 80% produced by this implementation. On the other hand, for $\mathrm{P} = (6, 1, 0)$, all twenty of Knuth's experiments produced matroids of rank 3, while in this case 13.8% of the matroids were of rank 4 or 5.

There are more statistical outliers with regards to rank (as expected, given the higher sample size), however the data still reflects Knuth's observation that

in most cases, the final rank is reduced by one for each enlargement made. As mentioned above, passing no enlargements to KNUTH-MATROID gives the uniform matroid, so RANDOM-KNUTH-MATROID$(n, (0, 0, 0)) = U_n$. The rank of the uniform matroid over the ground set E is $|E|$. As we can see in the results, RANDOM-KNUTH-MATROID$(10, (6, 0, 0))$ gives matroids of rank 4 in 91.7% of the cases, while RANDOM-KNUTH-MATROID$(10, (5, 2, 0))$ gives matroids of rank 3 in 79.1% of the cases.

When it comes to the number of bases, there are more significant discrepancies between the results given in this report and the ones in Knuth's paper. For the matroids generated with P $= (6, 0, 0)$ where the final rank was 4, my implementation produced on average 23.3% more bases than Knuth's implementation, whereas for P $= (5, 2, 0)$ and $r = 4$, we get 28.8% less bases on average. Limited to the hardware capabilities of the day, Knuth did relatively few experiments for each parameter setting, so this might be due to sample error.

Table 3.1: Observed mean values for RANDOM-KNUTH-MATROID.

n	$(p_1, p_2, \ldots)$	Trials	Bases	$\lvert F_2\rvert$	$\lvert F_3\rvert$	$\lvert F_4\rvert$	$\lvert F_5\rvert$	$\lvert F_6\rvert$
10	$(6,0,0)$	44 [a]	100.0	30.3	1.0			
10	$(6,0,0)$	917 [b]	76.6	28.3	25.5	1.0		
10	$(6,0,0)$	39 [c]	51.6	31.0	38.5	27.8	1.0	
10	$(5,1,0)$	26 [a]	107.2	33.3	1.0			
10	$(5,1,0)$	935 [b]	102.6	32.7	33.0	1.0		
10	$(5,1,0)$	39 [c]	53.0	33.0	44.6	48.0	1.0	
10	$(5,2,0)$	791 [a]	108.0	32.5	1.0			
10	$(5,2,0)$	201 [b]	100.0	32.9	32.6	1.0		
10	$(5,2,0)$	8 [c]	24.6	30.1	39.9	66.0	1.0	
10	$(6,1,0)$	862 [a]	99.2	28.4	1.0			
10	$(6,1,0)$	137 [b]	69.8	28.1	29.1	1.0		
10	$(6,1,0)$	1 [c]	48.0	33.0	41.0	33.0	1.0	
10	$(4,2,0)$	12 [a]	111.1	36.3	1.0			
10	$(4,2,0)$	950 [b]	119.2	35.9	42.5	1.0		
10	$(4,2,0)$	38 [c]	73.4	36.4	52.6	39.4	1.0	
10	$(3,3,0)$	4 [a]	115.0	39.0	1.0			
10	$(3,3,0)$	911 [b]	138.0	38.5	53.3	1.0		
10	$(3,3,0)$	85 [c]	90.6	38.7	61.9	36.2	1.0	
10	$(0,6,0)$	767 [b]	171.8	45.0	85.6	1.0		
10	$(0,6,0)$	230 [c]	128.4	45.0	95.8	72.7	1.0	
10	$(0,6,0)$	3 [d]	52.3	45.0	94.7	90.3	32.7	1.0

[a] Averages for experiments when final rank was 3.
[b] Averages for experiments when final rank was 4.
[c] Averages for experiments when final rank was 5.
[d] Averages for experiments when final rank was 6.

Chapter 4

Further work

Chapter 5

Conclusion

Although a conclusion may review the main points of the paper, it must not replicate the abstract. A conclusion might elaborate on the importance of the work or suggest applications and extensions. Do not cite references in the conclusion. Note that the conclusion section is the last section of the paper to be numbered. The appendix (if present), other acknowledgments, and references are listed without numbers.

Notes

1. TODO: Circuits
2. TODO: prate litt om typer matroider
3. TODO: presentere yankee swap etc, diskutere deres krav til Matroids.jl
4. TODO: Skrive konklusjonen. Gjøres til slutt

Bibliography

[1] Siddharth Barman and Arpita Biswas. Fair division under cardinality constraints, 2018.

[2] Donald E. Knuth. Random matroids. *Discrete Mathematics*, 12:341–358, 1975.