NTNU

Department of Computer Science

TDT4900 — Master's thesis

Exploring Matroids in Fair Allocation: Building the Matroids.jl Library

Author:
Andreas Aaberge Eide

Supervisor:
Magnus Lie Hetland

May 31, 2023

Contents

Chapter 1

Introduction

Imagine you are a dean at a large university, responsible for allocating seats in courses to students for the coming semester. Each student has applied for some subset of the courses available, and each course has a limit on the number of students it can accept. In addition, there are scheduling conflicts between courses, and the students have hard limits on the number of courses they can take in a semester[1]. As a good dean, you wish to allocate fairly, ensuring that the students are happy with their courses and do not feel disfavored compared to the other students. At the same time, you strive for efficiency, so that every student gets enough courses to make the expected progress on their grade. How would you go about solving this problem? Over lunch, you discuss your quandary with a colleague from the computer science department who immediately assuages your fears by informing you that your situation is in fact an instance of indivisible fair allocation with matroid-rank valuations, a well-studied problem for which several algorithms exist! Eager to learn more, you ask your colleague to explain her cryptic remark.

What is fair allocation?

Fair allocation is the problem of fairly partitioning a set of resources (in this case, the courses) among agents (the students) with different preferences or valuations

[1] This example is due to Benabbou et al [1].

over these resources. This has been a hot topic of interest since antiquity (a 2000-year old allocation strategy can be found in the Talmud [2]), and remains so today. The mathematical study of fair allocation started with a seminal work by Steinhaus in 1948 [3], and for decades the focus was largely on the *divisible* case, in which the resources can be divided into arbitrary small pieces. In the divisible case, fair allocations always exist, and they can be computed efficiently [4]. In the dean's scenario, however, the course seats are *indivisible goods*. A fair allocation of indivisible goods is, depending on the measure of fairness, not always achievable; consider for example allocating a course with one seat between two students who both applied for it – there is no way of allocating the seat without one agent being unhappy.

Generally speaking, an allocation is measured against two justice criteria: *fairness* and *efficiency*. Fairness relates to the degree to which agents preceive the allocation as favoring other agents over themselves. One common way to describe the *fairness* of an allocation is with the concept of *envy-freeness*. Envy is defined as the degree to which an agent values another agent's received bundle of resources higher than their own. An allocation is envy-free if no agent envies another agent. In the trivial example above, the only envy-free allocation is the one in which no student receives the seat; while this is technically speaking fair, it is highly inefficient. Efficiency deals with maximizing some notion of resource utilization, or, equivalently, reducing waste. The perfectly fair allocation in which no one receives anything is rarely desirable for reasons of efficiency. Conversely, while the allocation in which one agent receives everything might be highly efficient in terms of the total sum of bundle values, it is obviously unfair. The task of the fair allocation algorithm, then, is to find some balance between these criteria.

How do matroids enter into this?

What the colleague from the computer science department noticed about the dean's problem, was that it was well-structured, in fact it is a textbook example of *matroid-rank valuations* in practice. Matroid rank functions (MRFs) are are a class of functions with properties that make them both easy to reason about and practically applicable in a setting such as fair allocation, and can equivalently be referred to as *binary submodular functions*. A submodular function is a set function that obeys the law of *diminishing returns* – as the size of the input set increases, the marginal value of a single additional good decreases. MRFs are the class of submodular functions with *binary marginals*, meaning that the

value of any single good is either 0 or 1.

In practice, these properties make MRFs a compelling framework for modelling user preferences in a setting such as the dean's allocation scenario. The binary marginals reflect a student's willingness (value of 1) or unwillingness (value of 0) to enroll in a course. The diminishing returns property allow us to implement what are known in economics terms as *supplementary goods* and *fixed demand*. A student might be interested in two similar courses, but not wish to enroll in both, so given one, the marginal value of the other drops to 0 (the courses are supplementary goods). In addition, a student needs only at most one seat per course, so the many available seats for the same course are also supplementary goods. A student has limited time and energy, and so for each course seat received, the marginal value of the other courses can only decrease – after some threshold is reached in the number of enrolled courses, all remaining courses have value 0 (there is a fixed demand for courses).

Matroids are extensively studied mathematical structures that generalize concepts from a variety of different fields. A number of interesting algorithms have been developed for fair allocation with matroid-rank valuations [1, 5–8] that make use of deep results from matroid theory in their analysis, and deliver well on a range of justice criteria which might be computationally intractable to achieve under general valuations.

What does this thesis contribute?

Perhaps because matroids are so well-understood and pleasant to work with theoretically, there is a dearth of tooling available for generating and working with them programmatically. In an effort to fill this gap, this thesis proposes Matroids.jl, a library for the Julia programming language [9], which extends the existing Allocations.jl library [10] with the functionality required to enable the empirical study of matroidal fair allocation algorithms.

This thesis describes the work that has been done to design and build a working, proof-of-concept version of Matroids.jl. It is structured as follows. In the next chapter, I establish the concepts from matroid theory and fair allocation necessary to understand the rest of the thesis. In Chapter 3, I describe the design and implementation of the Matroids.jl API, which includes various classic matroid algorithms that have found use in fair allocation algorithms. In Chapter 4, I show how this API can be used to implement Viswanathan and Zick's Yankee Swap algorithm [8] and some other algorithms for matroid-rank-valued fair allocation. In Chapter 5, I show how Matroids.jl implements the

random generation of a range of matroid types. Of particular interest here is Knuth's classic method for generating arbitrary matroids [11], the successful implementation of which was a significant sub-goal of the project. In Chapter 6, I provide some experimental results for the algorithms over different matroid types. Finally, in Chapter 7, I give a summary discussion on the limitations of Matroids.jl and suggests a few possible avenues of future work.

Chapter 2

Preliminaries

> For simplicity, we also assume that every point in a geometry is a closed set. Without this additional assumption, the resulting structure is often described by the ineffably cacaphonic term "matroid", which we prefer to avoid in favor of the term "pregeometry".
>
> Gian-Carlo Rota [12]

Matroids were first introduced by Hassler Whitney in 1935 [13], in a seminal paper where he described two axioms for independence in the columns of a matrix, and defined any system obeying these axioms to be a "matroid" (which unfortunately for Rota is the term that has stuck). Whitney's key insight was that this abstraction of "independence" is applicable to both matrices and graphs. Matroids have also received attention from researchers in fair allocation, as their properties make them useful for modeling user preferences. We have already seen this with the course allocation problem described in the previous chapter; other use cases include the assignment of kindergarten slots or public housing estates among people of different ethnicities [7].

2.1 Fair allocation

In this thesis, I use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. To ease readability, I abuse notation a bit and replace $A \cup \{g\}$ and $A \setminus \{g\}$ with $A + g$ and $A - g$, respectively.

An instance of a fair allocation problem consists of a set of agents $\mathcal{N} = [n]$ and a set of m goods $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$.Each agent has a valuation function $v_i : 2^{\mathcal{G}} \to \mathbb{R}^+$; $v_i(A)$ is the value agent i ascribes to the bundle of goods A. The marginal value of agent i for the good g, given that she already owns the bundle A, is given by $\Delta_i(A, g) := v_i(A+g) - v_i(A)$. Throughout most of this thesis, we assume that v_i is a matroid rank function, or, equivalently, a binary submodular function. To formalize the description given in Chapter 1, this means that

(a) $v_i(\emptyset) = 0$,

(b) v_i has binary marginals: $\Delta_i(A, g) \in \{0, 1\}$ for every $A \subset \mathcal{G}$ and $g \in \mathcal{G}$

(c) v_i is submodular: for every $A \subseteq B \subseteq \mathcal{G}$ and $g \in \mathcal{G} \setminus B$, we have that $\Delta_i(A, g) \geq \Delta_i(B, g)$.

Any function v_i adhering to these properties precisely determine a matroid [14]. There are many other ways to characterize matroids, some of which are given in Section 2.2.

The output of an algorithm for fair allocation is an allocation of the goods to the agents. An allocation A is an n-partition of E, $A = (A_1, A_2, \ldots, A_n)$, where each A_i is the bundle of goods allocated to agent i. Sharing is not allowed, so $A_i \cap A_j = \emptyset$ for all $i \neq j$. We say that an allocation is *clean* if no agent has received any good they value at 0. An allocation is *complete* if all goods are allocated.

2.1.1 Envy-freeness

We are interested in producing *fair* allocations. One of the most popular notions of fairness in the literature is envy-freeness (EF), which states that no agent should prefer another agent's bundle over her own. An allocation A is EF iff, for all agents $i, j \in \mathcal{N}$,

$$v_i(A_i) \geq v_i(A_j). \tag{EF}$$

Because, as mentioned in the introduction, EF is not always achievable when the goods are indivisible, the literature has focused on relaxations thereof. The most prominent such relaxation, which can be guaranteed, is *envy-freeness up*

to one good (EF1) [15], which allows for the envy of up to the value of one (highest-valued) good. A is an EF1 allocation iff, for all agents $i, j \in \mathcal{N}$, there exists a $g \in A_j$ such that

$$v_i(A_i) \geq v_i(A_j - g). \tag{EF1}$$

Envy-freeness up to any good (EFX) is an even stronger version of EF. While EF1 allows that agent i envies agent j up to their highest valued good, EFX requires that the envy can be removed by dropping agent j's least valued good. There are two slightly different definitions of EFX in use in the literature. I follow the naming scheme used by Benabbou et al. [7] and refer to these as EFX_+ and EFX_0. Caragiannis et al. [16] requires that this least valued good be positively valued. We call this fairness objective EFX_+. A is an EFX_+ allocation iff, for all agents $i, j \in \mathcal{N}$,

$$v_i(A_i) \geq v_i(A_j - g),\ \forall g \in A_j \text{ st. } v_i(A_j - g) < v_i(A_j). \tag{EFX$_+$}$$

Plaut and Roughgarden [17], on the other hand, allow for 0-valued goods in the envy check – we call this version EFX_0. It is stronger requirement than EFX_+. A is an EFX_0 allocation iff, for all agents $i, j \in \mathcal{N}$,

$$v_i(A_i) \geq v_i(A_j - g),\ \forall g \in A_j. \tag{EFX$_0$}$$

2.1.2 Proportionality

Proportionality is a fairness objective that is fundamentally different from envy-freeness, in that it checks each bundle value against some threshold, instead of comparing bundle values against each other. An allocation A is proportional (PROP) iff each agent $i \in \mathcal{N}$ receives at least her proportional share PROP_i, which is the $\frac{1}{n}$ fraction of the value she puts on the whole set of goods, ie.

$$v_i(A_i) \geq \text{PROP}_i := \frac{v_i(\mathcal{G})}{n}. \tag{PROP}$$

Proportionality might not be achievable in the indivisible case (again, consider two agents and one positively valued good), and so relaxations in the same vein as EF1 and EFX have been introduced – these are called PROP1 and PROPX [4]. An allocation is PROP1 iff there for each agent $i \in \mathcal{N}$ exists some good $g \in \mathcal{G} \setminus A_i$ that, if given to i, would ensure that agent i received her proportional share; that is,

$$\exists g \in \mathcal{G} \setminus A_i \text{ st. } v_i(A_i + g) \geq \text{PROP}_i \tag{PROP1}$$

PROPX is a stronger fairness objective than PROP1, and has, as in the case of EFX, two slightly different definitions in the literature. I follow the naming scheme established for EFX above, and refer to these as PROPX_+ and PROPX_0. The logic is similar to that of EFX. An allocation is PROPX_0 iff each agent can achieve her proportional share by receiving one additional, least-valued good from the goods not allocated to her. This good might be zero-valued.

$$\min_{g \in \mathcal{G} \setminus A_i} v_i(A_i + g) \geq \text{PROP}_i \tag{PROPX$_0$}$$

If we disallow zero-valued items, we arrive at the even stronger criteria PROPX_+, given by:

$$\min_{g \in \mathcal{G} \setminus A_i, \Delta_i(A_i, g) > 0} v_i(A_i + g) \geq \text{PROP}_i \tag{PROPX$_+$}$$

Maximin share fairness

Budish introduces a relaxation of proportionality known as *maximin share fairness* (MMS) [18], in which the threshold for each agent i is her *maximin share*, defined as the maximum value she could receive if she partitioned $\mathcal{G}$ among all agents and then picked the worst bundle. Let $\mathcal{A}_n(\mathcal{G})$ be the family of all possible allocations of the goods in $\mathcal{G}$ to the agents in $\mathcal{N}$. The maximin share for agent i, denoted by μ_i, is given by

$$\mu_i := \max_{B \in \mathcal{A}_n(\mathcal{G})} \min_{A \in B} v_i(A).$$

An allocation is *maximin share fair* (MMS) if all agents receive at least as much as their maximin share:

$$v_i(A_i) \geq \mu_i, \ \forall i \in \mathcal{N}. \tag{MMS}$$

2.1.3 Efficiency

As mentioned in the introduction, fairness is usually coupled with some efficiency criterion, to prevent the perfectly fair solution in which the whole set of goods is thrown away. The efficiency of an allocation can be measured with some *welfare function* on the values of the agents. I enumerate the three most commonly discussed in the literature:

1. **Egalitarian social welfare (ESW):** The ESW of an allocation A is given by the minimum value of an agent. $\text{ESW}(A) = \min_{i \in \mathcal{N}} v_i(A_i)$.

2. **Utilitarian social welfare (USW):** The USW of an allocation is given by the sum of agent values. $\text{USW}(A) = \sum_{i \in \mathcal{N}} v_i(A_i)$.

3. **Nash welfare (NW):** The Nash welfare of an allocation is a compromise between the utilitarian and egalitarian approaches, given by the product of agent utilities. $\text{NW}(A) = \prod_{i \in \mathcal{N}} v_i(A_i)$.

Allocations that maximize one of these welfare functions are referred to as MAX-ESW, MAX-USW and MNW, respectively.

Leximin

An obvious drawback of the egalitarian rule is that, in the situation where there are multiple possible allocations that maximize the minimum bundle value, it is indifferent to which of these bundles it prefers. Consider for instance two possible allocations of goods to three agents, where their bundle values are given as (A,B,C):

- Allocation 1: $(5, 7, 10)$
- Allocation 2: $(5, 8, 9)$

Both of these allocations are MAX-ESW. A stricter version of the egalitarian rule is *leximin* – an allocation is leximin if it maximizes the smallest value; subject to that, it maximizes the second-smallest value; subject to that, it maximizes the next-smallest value, and so on. The leximin rule prefers Allocation 2 over Allocation 1.

2.2 Matroid theory

If a mathematical structure can be defined or axiomatized in multiple different, but not obviously equivalent, ways, the different definitions or axiomatizations of that structure make up a cryptomorphism. The many obtusely equivalent definitions of a matroid are a classic example of cryptomorphism, and belie the fact that the matroid is a generalization of concepts in many, seemingly disparate areas of mathematics. As a result of this, the terms used in matroid theory are borrowed from analogous concepts in both graph theory and linear algebra.

The most common way to characterize a matroid is as an *independence system*. An independence system is a pair $(E, \mathcal{I})$, where E is the ground set of

elements, $E \neq \emptyset$, and $\mathcal{I}$ is the set of independent sets, $\mathcal{I} \subseteq 2^E$. The *dependent sets* of a matroid are $2^E \setminus \mathcal{I}$.

In practice, the ground set E represents the universe of elements in play, and the independent sets of typically represent the legal combinations of these items. In the context of fair allocation, the independent sets represent the legal (in the case of matroid constraints) or desired (in the case of matroid utilities) bundles of items.

A matroid is an independence system with the following properties [13]:

(1) If $A \subseteq B$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$.

(2) If $A, B \in \mathcal{I}$ and $|A| > |B|$, then there exists $e \in A - B$ such that $B + e \in S$.

(2') If $S \subseteq E$, then the maximal independent subsets of S are equal in size.

Property (1) is called the *hereditary property* and (2) the *exchange property*. Properties (2) and (2') are equivalent. To see that (2) $\implies$ (2'), consider two maximal subsets of S. If they differ in size, (2) tells us that there are elements we can add from one to the other until they have equal cardinality. We get (2') $\implies$ (2) by considering $S = A \cup B$. Since $|A| > |B|$, they cannot both be maximal, and some $e \in A \setminus B$ can be added to B to obtain another independent set.

When $S = E$, (2') gives us that the maximal independent sets of a matroid are all of the same size. A maximal independent subset of E is known as a *basis*. The size of the bases is the *rank* of the matroid as a whole. A matroid can be exactly determined by $\mathcal{B}$, its collection of bases, since a set is independent if and only if it is contained in a basis (this follows from (1) above). A theorem by Whitney [13] gives the axiom system characterizing a collection of bases of a matroid:

1. No proper subset of a basis is a basis.
2. If $B, B' \in \mathcal{B}$ and $e \in B$, then for some $e' \in B'$, $B - e + e' \in \mathcal{B}$.

The rank function of a matroid is a function $v : 2^E \to \mathbb{Z}^+$ which, given a subset $B \subseteq E$, returns the size of the largest independent set contained in B. That is,

$$v(B) = \max_{A \subseteq B, A \in \mathcal{I}} |A|.$$

2.2.1 A graphic example

Different types of matroids exist, arising from different sources of "independence"; one well-known subclass of matroids, arising from notions of independence in graphs, is the class of *graphic matroids*. Graphic matroids are by their nature more visualizable than other types of matroids, and will be used to

A *tree* is a connected acyclic graph, and a *forest* is a disconnected graph consisting of some number of trees. A *spanning tree* of G is a subgraph with a unique simple path between all pairs of vertices of G. A *spanning forest* of G is a collection of spanning trees, one for each component. A graph will have some number of different spanning trees. Figure 2.1 shows two spanning trees of the same graphs (or alternatively, a spanning forest over one graph with two components).

Given a graph $G = (V, E)$, let $\mathcal{I} \subseteq 2^E$ be the family of subsets of the edges E such that, for each $I \in \mathcal{I}$, (V, I) is a forest. It is a classic result of matroid theory that $\mathfrak{M} = (E, \mathcal{I})$ (the ground set of the matroid being the edges of the graph) is a matroid [14, p. 657]. To understand how, we will show that it adhers to axioms (1) and (2'), as given in above. By investigating the highlighted spanning trees in Figure 2.1, it is easy to convince oneself that all subsets of a spanning tree are trees, since no subset of an acyclic set of edges will contain a cycle. Thus axiom (1) – the hereditary property – holds.

To see that (2') holds, consider the set of bases of the matroid, $\mathcal{B} \subseteq \mathcal{I}$. Figure 2.1 shows two bases of the matroid described by the graph. By definition, each basis $B \in \mathcal{B}$ is a maximal forest over G. Since a spanning tree of a graph with n nodes must needs have $n - 1$ edges (I recommend drawing trees and counting their edges until one is convinced that this must be the case), we have $|B| = |V| - k$, where k is the number of components of G. This is the same for every $B \in \mathcal{B}$, which proves property (2'). Any matroid given by a graph G, denoted by $\mathfrak{M}(G)$, is called a graphic matroid.

2.2.2 Other characterizations of a matroid

We have already seen the independent sets characterization of a matroid, but the other properties of a matroid have their own axiom systems that can equivalently be used to characterize a matroid.

Characterization via circuits. A *circuit* is a minimal dependent set of a matroid – it is an independent set plus one "redundant" element. Equivalently, the collection of circuits of a matroid is given by

$$\mathcal{C} = \big\{C : |C| = r(C) + 1, C \subseteq E\big\}.$$

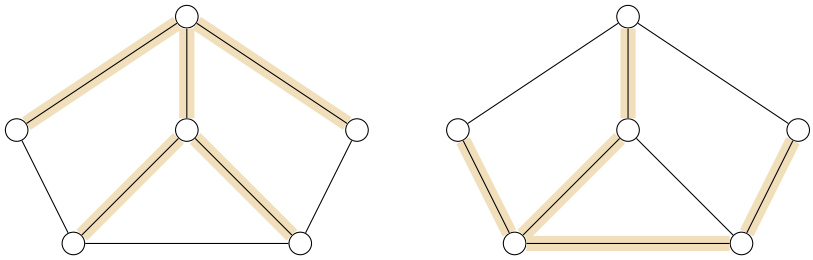

Figure 2.1: Two spanning trees of a graph with 8 edges.

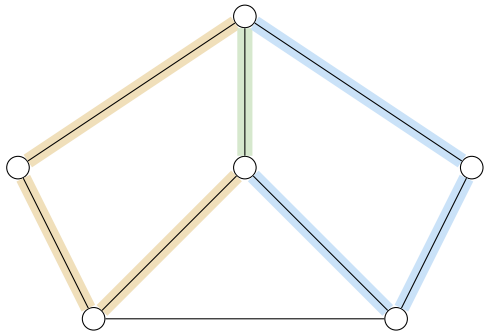

Figure 2.2: Two overlapping circuits on a graph.

A set is independent if and only if it contains no circuit [14], and so a matroid is uniquely determined by the collection of its circuits. The following conditions characterize $\mathcal{C}$ [13]:

(1) No proper subset of a circuit is a circuit.

(2) If $C, C' \in \mathcal{C}$, $x \in C \cap C'$ and $y \in C \setminus C'$, then $C \cup C'$ contains a circuit containing y but not x.

These properties are easily grokked by studying an example. In Figure 2.2, we see two circuits of a graph, highlighted in yellow and blue, overlapping at the edge highlighted in green. Obviously, no circuit in this figure contains a circuit. We can also verify that the union of the yellow edge set and the blue edge set, minus the green edge, is in fact a third, bigger circuit, as promised by property (2) above.

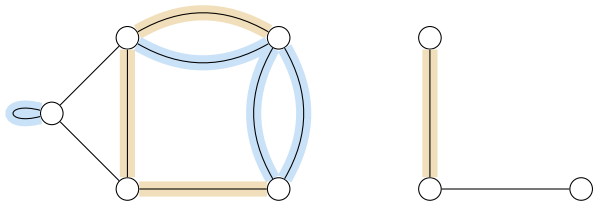

Figure 2.3: An independent subset of edges in yellow, with its closure in blue

Characterization via closed sets. We also need to establish the concept of the *closed sets* (sometimes referred to as *flats* [14]) of a matroid. A closed set is a set whose cardinality is maximal for its rank. Equivalently to the definition given above, we can define a matroid as $\mathfrak{M} = (E, \mathcal{F})$, where $\mathcal{F}$ is the set of closed sets of $\mathfrak{M}$, satisfying the following properties [11]:

1. The set of all elements is closed: $E \in \mathcal{F}$
2. The intersection of two closed sets is a closed set: If $A, B \in \mathcal{F}$, then $A \cap B \in \mathcal{F}$
3. 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 $A \cup \{b\}$

The *closure function* is the function $cl : 2^E \to 2^E$, such that

$$cl(S) = \big\{x \in E : r(S) = r(S \cup \{x\})\big\}.$$

That is to say, the closure function, when given a set $S \subseteq E$, returns the set of elements in $x \in E$ such that x can be added to S with no increase in rank. It returns the closed set of the same rank as S, that contains S. The *nullity* of a subset S is the difference $|S| - r(S)$, ie. the number of elements that must be removed from S to obtain an independent set.

Figure 2.3 shows an independent (acyclic) set S of edges in highlighted yellow, along with its closure $cl(S)$ in blue. The closure is the set of edges e such that $v(S + e) = v(S)$, or, equivalently, such that the spanning tree of $S + e$ has the same size as that of S.

2.2.3 Matroid union

The matroid union is an operation that allows us to produce a new matroid by combining the independent sets of some number of existing matroids. Given n

matroids $\mathfrak{M}_i = (E, \mathcal{I}_i)$, their union is given by

$$\widehat{\mathfrak{M}} = (E, \{I_1 \cup \ldots \cup I_n : I_i \in \mathcal{I}_i,\ \forall i \in [n]\}).$$

$\widehat{\mathfrak{M}}$ is in fact a matroid [14, Ch. 42], whose independent sets are the the subsets $S \subseteq E$ that allow an n-partition $S_1, \ldots, S_n$ such that $S_i \in \mathcal{I}_i$, for all $i \in [n]$. In fair allocation jargon, if each $\mathfrak{M}_i$ is the matroid described by agent i's valuation function v_i, a set of goods S is independent in $\widehat{\mathfrak{M}}$ if and only if we can allocate it among the agents and produce utilitarian social welfare equal to $|S|$. Hence, each basis in $\widehat{\mathfrak{M}}$ corresponds to a clean (but not necessarily complete), MAX-USW allocation of the goods in $\mathcal{M}$ [6]. It is a classic result of Edmonds [19] that a basis in $\widehat{\mathfrak{M}}$ can be computed in polynomial time, using the *matroid union algorithm*.

2.2.4 Matroid erection

Knuth's general matroid construction, the implementation of which is discussed in Section 5.4, requires us to establish a few more matroid concepts. The *rank-k truncation* of a matroid $\mathfrak{M} = (E, \mathcal{I})$, is the matroid $\mathfrak{M}^{(k)} = (E, \mathcal{I}^{(k)})$, where

$$\mathcal{I}^{(k)} = \{I \in \mathcal{I} : |I| \leq k\}.$$

When $\mathfrak{M}$ is of rank r, its *truncation* is given as $T(\mathfrak{M}) = \mathfrak{M}^{(r-1)}$. As a simple example, we have that the uniform matroid $U_n^{n-1} = T(\mathfrak{F}_n)$, where $\mathfrak{F}_n$ is the free matroid with n elements. The *erection* (I refer to Crapo for the choice of term [20]) of the matroid $\mathfrak{M}$ is the matroid $\mathfrak{N}$ such that $\mathfrak{M} = T(\mathfrak{N})$ [20]. A matroid can have many erections. For example, it is easy to see that U_n^4 is an erection of U_n^3, since U_n^4 by definition has all the same independent sets of the rank-3 matroid U_n^3, along with all size-4 subsets of E. However, let $\mathfrak{U}$ be the rank-4 matroid over E where every subset of E of rank ≤ 3 is independent, and which has one size-4 rank-3 dependent set. The rank-3 independent sets of $\mathfrak{U}$ are the same as those of U_n^3, so we have $U_n^3 = T(\mathfrak{U})$, however the introduction of a size-4 dependent set of rank 3 has erected a different rank-4 matroid.

2.3 Matroids in fair allocation

It should be clear at this point that matroids are compelling structures to work with in the context of fair allocations. There are two main use cases for matroids in fair allocation: matroid-rank valuation functions (as in the example scenario

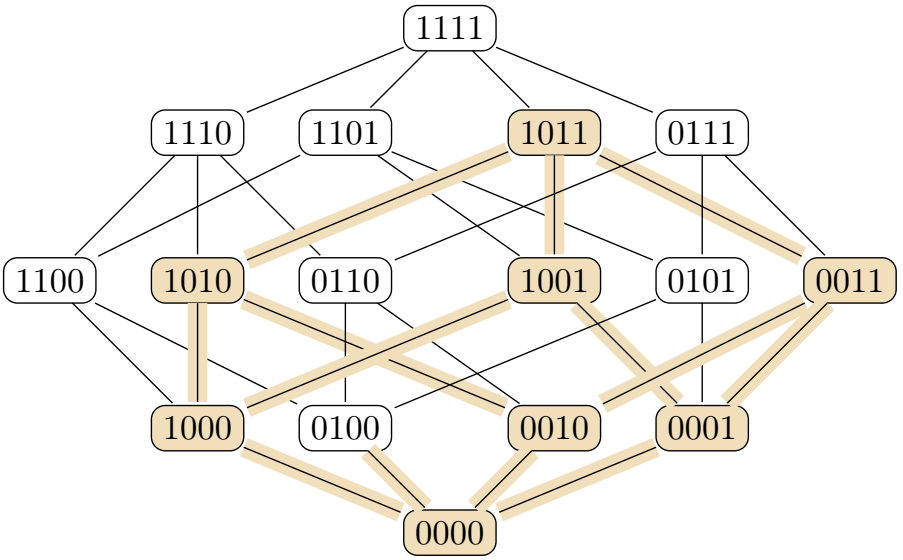

Figure 2.4: The caption

from the introduction) and matroid constraints. Matroids.jl will be developed with the empirical study of algorithms for these two scenarios in mind.

2.3.1 Matroid-rank valuations

In a fair allocation instance with matroid-rank valuations, each agent i has a corresponding matroid $\mathfrak{M}_i = (\mathcal{M}, \mathcal{I}_i)$, where $\mathcal{M}$ (the set of goods) is the ground set of elements common to all agents' matroids.

Babaioff 2021 – PE mechanism – non-redundant Lorenz dominating allocation always exists and is MNW, MAX-USW, leximin, EFX (EF1) and .5-MMS. With a randomized prioritization it is ex ante EF and ex ante proportional as well. 3Yankee Swap computes prioritized non-redundant Lorenz dominating allocations.

Limitation: MRFs cannot model complementary goods!

2.3.2 Matroid constraints

Another usage found for matroids in fair allocation is that of *matroid constraints*. The majority of work on fair division assumes that any allocation is feasible, and the sole concern is finding an allocation that aligns well with the agents' valuation profiles. In many practical applications, however, there will be allocations that are not legal or desirable. Suksompong [21] gives the example of a museum with multiple branches distributing exhibits of different categories (sculpture, paintings, et cetera) among the branches. For each category, it wants to create a balanced distribution among the branches, so that the difference in the number of exhibits of a given category differ by at most one between any branch. This is an example of a *cardinality constraint*, which can be modeled with a partition matroid, and is a subset of the broader class of matroid constraints.

When matroid constraints are enforced on a fair allocation instance, we require that all bundles be independent sets on some supplied matroid, common to all agents.

[8] V. Viswanathan and Y. Zick, *Yankee swap: A fast and simple fair allocation mechanism for matroid rank valuations*, 2023. arXiv: 2206.08495 [cs.DS].

[9] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017. [Online]. Available: https://doi.org/10.1137/141000671.

[10] M. L. Hetland and H. Hummel, *Allocations.jl*, version 0.1, Nov. 2022. [Online]. Available: https://github.com/mlhetland/Allocations.jl.

[11] D. E. Knuth, "Random matroids," *Discrete Mathematics*, vol. 12, pp. 341–358, 4 1975.

[12] H. H. Crapo and G.-C. Rota, *On the foundations of combinatorial theory: Combinatorial geometries*. M.I.T. Press, 1970.

[13] H. Whitney, "On the abstract properties of linear dependence," *American Journal of Mathematics*, vol. 57, pp. 509–533, 3 Jul. 1935.

[14] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

[15] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi, "On approximately fair allocations of indivisible goods," in *Proceedings of the 5th ACM Conference on Electronic Commerce*, ser. EC '04, New York, NY, USA: Association for Computing Machinery, 2004, pp. 125–131, ISBN: 1581137710. DOI: 10.1145/988772.988792. [Online]. Available: https://doi.org/10.1145/988772.988792.

[16] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *ACM Trans. Econ. Comput.*, vol. 7, no. 3, Sep. 2019, ISSN: 2167-8375. DOI: 10.1145/3355902. [Online]. Available: https://doi.org/10.1145/3355902.

[17] B. Plaut and T. Roughgarden, *Almost envy-freeness with general valuations*, 2017. arXiv: 1707.04769 [cs.GT].

[18] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, vol. 119, no. 6, pp. 1061–1103, Dec. 2011. DOI: 10.1086/664613. [Online]. Available: https://doi.org/10.1086/664613.

[19] J. Edmonds, "Matroid partition," in *50 Years of Integer Programming 1958-2008*, Springer Berlin Heidelberg, Nov. 2009, pp. 199–217. DOI: 10.1007/978-3-540-68279-0_7. [Online]. Available: https://doi.org/10.1007/978-3-540-68279-0_7.

[20] H. H. Crapo, "Erecting geometries," *Annals of the New York Academy of Sciences*, vol. 175, no. 1, pp. 89–92, Jul. 1970. DOI: 10.1111/j.1749-6632.1970.tb56458.x. [Online]. Available: https://doi.org/10.1111/j.1749-6632.1970.tb56458.x.

[21] W. Suksompong, "Constraints in fair division," *SIGecom Exch.*, vol. 19, no. 2, pp. 46–61, Dec. 2021. DOI: 10.1145/3505156.3505162. [Online]. Available: https://doi.org/10.1145/3505156.3505162.

[22] S. E. Fienberg, "A brief history of statistical models for network analysis and open challenges," *Journal of Computational and Graphical Statistics*, vol. 21, no. 4, pp. 825–839, 2012. DOI: 10.1080/10618600.2012.738106. eprint: https://doi.org/10.1080/10618600.2012.738106. [Online]. Available: https://doi.org/10.1080/10618600.2012.738106.

[23] P. Erdős and A. Rényi, "On random graphs I.," *Publicationes Mathematicae Debrecen*, vol. 6, no. 3-4, pp. 290–297, Jul. 1959. DOI: 10.5486/pmd.1959.6.3-4.12. [Online]. Available: https://doi.org/10.5486/pmd.1959.6.3-4.12.

[24] E. N. Gilbert, "Random Graphs," *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, 1959. DOI: 10.1214/aoms/1177706098. [Online]. Available: https://doi.org/10.1214/aoms/1177706098.

[25] S. Janson, D. E. Knuth, T. Łuczak, and B. Pittel, *The birth of the giant component*, 1993. arXiv: math/9310236 [math.PR].

[26] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, 1 Jan. 2002. DOI: 10.1103/RevModPhys.74.47. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.47.

[27] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998. DOI: 10.1038/30918. [Online]. Available: https://doi.org/10.1038/30918.

[28] J. Fairbanks, M. Besançon, S. Simon, J. Hoffiman, N. Eubank, and S. Karpinski, *JuliaGraphs/Graphs.jl: An optimized graphs package for the julia programming language*, 2021. [Online]. Available: https://github.com/JuliaGraphs/Graphs.jl/.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (The MIT Press), 3rd ed. London, England: MIT Press, Jul. 2009.

[30] N. Wirth and C. A. R. Hoare, "A contribution to the development of ALGOL," *Commun. ACM*, vol. 9, no. 6, pp. 413–432, Jun. 1966, ISSN: 0001-0782. DOI: 10.1145/365696.365702. [Online]. Available: https://doi.org/10.1145/365696.365702.

[31] D. E. Knuth, *ERECTION.W*, https://www-cs-faculty.stanford.edu/~knuth/programs/erection.w, Mar. 2003. [Online]. Available: https://www-cs-faculty.stanford.edu/~knuth/programs/erection.w.

[32] T. Greene, "Descriptively sufficient subcollections of flats in matroids," *Discrete Mathematics*, vol. 87, no. 2, pp. 149–161, 1991, ISSN: 0012-365X. DOI: https://doi.org/10.1016/0012-365X(91)90044-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X91900443.

[33] R. Fourquet, *BitIntegers.jl*, https://github.com/rfourquet/BitIntegers.jl, Accessed: 2023-05-20.