A Gentle Introduction to Matroid Algorithmics

Matthias F. Stallmann

April 14, 2016

1 Matroid axioms

The term *matroid* was first used in 1935 by Hassler Whitney [34]. Most of the material in this overview comes from the textbooks of Lawler [25] and Welsh [33].

A matroid is defined by a set of axioms with respect to a ground set S. There are several ways to do this. Among these are:

Independent sets. Let $\mathcal{I}$ be a collection of subsets defined by

- (I1) $\emptyset \in \mathcal{I}$
- (I2) if $X \in \mathcal{I}$ and $Y \subseteq X$ then $Y \in \mathcal{I}$
- (I3) if $X, Y \in \mathcal{I}$ with $|X| = |Y| + 1$ then there exists $x \in X \setminus Y$ such that $Y \cup x \in \mathcal{I}$

A classic example is a *graphic matroid*. Here S is the set of edges of a graph and $X \in \mathcal{I}$ if X is a forest. (I1) and (I2) are clearly satisfied. (I3) is a simplified version of the key ingredient in the correctness proofs of most minimum spanning tree algorithms. If a set of edges X forms a forest then $|V(X)| - |X| = |c(X)|$, where $V(x)$ is the set of endpoints of X and $c(X)$ is the set of components induced by X. If $|X| = |Y| + 1$, then $|c(X)| = |c(Y)| - 1$, meaning X must have at least one edge x that connects two of the components induced by Y. So $Y \cup x$ is also a forest.

Another classic algorithmic example is the following uniprocessor scheduling problem, let's call it the *scheduling matroid*. Let S be a set of tasks indexed $1, \ldots, n$ where each task i takes unit time and has an integer deadline d_i. A subset X of tasks is in $\mathcal{I}$ if they can all be completed by their deadlines. In this setting, there are no precedence constraints among the tasks. Aside: the variation where tasks have a release time before which they cannot be executed also induces a slightly more general matroid.

Here again, (I1) and (I2) are trivial. To see that (I3) is true, let i be a task in $X \setminus Y$ with d_i as large as possible. If $d_i > d_j$ for all $j \in Y$ we are done: we can schedule i after all tasks in Y are finished. Otherwise X has at most d_i tasks with deadlines $\leq d_i$ and at most $d_i - 1$ of these are in Y (otherwise $d_i \in Y$). So we can schedule task i in slot d_i (the slot right before deadline d_i).

Another axiomatic description of a matroid is in terms of bases. Here $\mathcal{B}$ is a collection of subsets of S satisfying the following.

- (B1) if $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$ then there exists $y \in B_2 \setminus B_1$ such that $(B_1 \cup y) \setminus x \in \mathcal{B}$.

A base in a graphic matroid is a spanning forest. In the scheduling matroid it is a maximal set of tasks that can all be scheduled before their deadlines.

Equivalence of these axioms sets can be proved for special cases but can also be proved abstractly.

The following is easy to prove from (I3).

Theorem 1 *If X and Y are independent and $|X| > |Y|$ then there exists $Z \subseteq X \setminus Y$ such that $|Y \cup Z| = |X|$ and $Y \cup Z$ is independent.*

It follows that ...

Corollary 1 *A maximal independent set of a matroid is also a* maximum *independent set. All bases of a matroid have the same cardinality.*

If we define a base to be a maximum independent set, then (B1) can be proved easily. Or we can start with bases and define an independent set to be a subset of a base.

The fact that all bases have the same cardinality leads to another axiomatic charaterization – in terms of *rank*, where rank is a function $\rho : 2^S \mapsto \mathcal{Z}$.

- (R1) $0 \leq \rho(X) \leq |X|$
- (R2) $X \subseteq Y$ implies $\rho(X) \leq \rho(Y)$
- (R3) $\rho(X \cup Y) + \rho(X \cap Y) \leq \rho(X) + \rho(Y)$ (submodularity)

The rank of a set X is the maximum cardinality of an independent subset of X. For example, if a graph $G = (V, E)$ is connected and $F \subseteq E$ then $\rho(F) = |V(F)| - |c(F)|$.

A final way (not quite – there are several others – but we'll stop here) to define matroids is in terms of circuits. Here $\mathcal{C}$ is a collection of subsets of S satisfying the following.

- (C1) if $C_1, C_2 \in \mathcal{C}$ and $C_1 \neq C_2$ then $C_1 \not\subseteq C_2$
- (C2) if $C_1, C_2 \in \mathcal{C}$, $C_1 \neq C_2$ and $z \in C_1 \cap C_2$, there exists $C_3 \in \mathcal{C}$ such that $C_3 \subseteq (C_1 \cup C_2) \setminus z$

In a graphic matroid a circuit is a simple cycle. In a scheduling matroid it is a set C of $k+1$ tasks, all of which have deadlines $\leq k$ and no subset of C has this property. In other words, a circuit C is a minimally dependent set: C is not independent but every subset of C is. It is easy to show that if C is a circuit then $\rho(C) = |C| - 1$.

Theorem 2 *If B is a base of $M(S)$ and $x \in S \setminus B$ then there exists a unique circuit $C(x, B)$ such that $x \in C(x, B) \subseteq B \cup x$. Furthermore, if $y \in C(x, B)$ then $(B \cup x) \setminus y$ is also a base.*

The circuit $C(x, B)$ is called the fundamental circuit of x with respect to the base B.

2 Matroid algorithmics

Matroids are important algorithmically because of the fact that a maximal independent set is also maximum. This means that one can use a greedy algorithm to find a minimum cost (minimum spanning tree) or maximum profit (maximum profit set of tasks to be scheduled) base of a matroid. It is useful first to define the concept of *contraction*.

Definition. The *contraction* of a matroid M with respect to an element e on ground set S, denoted as M/e is a matroid $M' = (S \setminus e, \mathcal{B}')$ such that if $B' \in \mathcal{B}'$, i.e., B' is base of M', then $B' \cup e$ is a base of M. This is a simplification of the formal definition of contraction in the textbooks [2, 33], but works well for our purposes.

In a graphic matroid contraction with respect to an edge $e = vw$ is the obvious operation: replace v and w with a new vertex x whose neighbors are those of v and w; parallel edges and self-loops are not an issue – the underlying graph of a graphic matroid need not be a simple graph.

In a scheduling matroid contraction with respect to task i with deadline d_i modifies each task j with $d_j \geq d_i$ so that it now has deadline $d'_j = d_j - 1$.

A simple recursive algorithm for finding the minimum cost base of a matroid, given a cost function f on the elements, is as follows (max profit is analogous).

```
function GREEDY-MIN-COST(M = (S, B)) is
    if rank of M is 0 (no independent sets) then return ∅
    else
        choose x ∈ S so that ρ(x) > 0 and f(x) is minimum
        return x ∪ GREEDY-MIN-COST(M/x)
    endif
end GREEDY-MIN-COST
```

Correctness follows directly from the definition of contraction and (B1): x must be included in an optimum base. Suppose base B is the minimum cost base that includes x. Let B' be any other base. If $x \notin B'$ then there exists $y \in B' \setminus B$ such that $B'' = (B' \cup x)\ y$ is also a base. Since $f(y) \geq f(x)$ we know that the cost of B'' is at least that of B.

Efficient greedy algorithms for both the graphic and the scheduling matroid are based on the fact that contraction can be implemented using disjoint set union. In the case of a graphic matroid the elements are vertices and, when an edge uv is contracted, the sets containing u and v are merged. In a scheduling matroid the elements are deadlines and when a contraction takes place with respect to a task with deadline d, the sets containing d and $d-1$ are merged (effectively changing tasks with deadlines d to ones with deadlines $d-1$). If matroid elements are already sorted by increasing cost/decreasing profit the time bound is therefore $O(m\alpha(m,n))$ where m is the number of elements and n is the rank (each contraction reduces the rank by one). In the special case of the scheduling matroid the arguments of the union operations are known in advance. Gabow and Tarjan [18] took advantage of this fact to reduce the time to $O(m)$; their algorithm makes heavy use of random access (to maintain tables indicating the unions that have been done) instead of simple pointer chasing.

Algorithm Greedy-Min-Cost, when specialized to graphic matroids, is Kruskal's algorithm for minimum spanning trees. Other algorithms are derived if we replace the requirement that x have minimum cost with the weaker one that x is guaranteed to be in a minimum cost base, usually proved using Theorem 2. The ability to derive almost linear time algorithms[1] for minimum spanning trees stems from the fact that contraction is an efficient, purely local, operation – see, e.g., Gabow et al. [14].

The best known time bound of a greedy algorithm for the scheduling matroid is $O(m+n\lg n)$, where $m=|S|$ and $n=\rho(s)$ (see, e.g., Gabow and Tarjan [17]). We can make three observations: (a) there is no need to consider deadlines $> m$; (b) for any deadline d, only the d largest profit tasks need to be considered; and (c) the highest profit task with deadline $\geq n$ is included in some base. Note that $n \leq \min(m, d_{\max})$ where $d_{\max}$ is the latest deadline.

The algorithm Max-Profit-Schedule is as follows.

1. For $d=1$ to m, create a (max) heap H_d of (the at most d highest profit) tasks having deadline d; include tasks with deadlines $> m$ in H_m.
2. Let $T=\emptyset$ (T is the set of tasks to be scheduled)
3. For $d=m$ down to 1:
   (a) let $i=\text{removeMax}(H_d)$
   (b) add i to T
   (c) merge H_d into H_{d-1}

The algorithm effectively does a the contraction when H_d is merged into H_{d-1}. Correctness follows from the fact that each iteration of the loop in step 3 adds a task of greatest profit that has deadline $\geq$ the rank of the current (possibly contracted) matroid.

Step 1 can be done in time $O(m)$: use a linear-time median algorithm to get the d highest profit tasks (when there are more than d tasks) for H_d. Any efficient mergeable heap implementation can create a heap in linear time. The number of removeMax operations in step 3 is n; each one adds an element to the base. These therefore take time $O(n\lg m)$. If Fibonacci heaps (or similar) are used, the merges can be done in linear time. Total time is $O(m+n\lg m)$.

[1]The *almost* part of this remark applies to purely combinatorial algorithms. There exist linear time algorithms that play around with the representations of the numbers.

3 Matroid taxonomy

Here we explore the different types of matroids that arise in practical applications and their theoretical underpinnings. Classes of matroids arise if we consider closure under duality and minors.

3.1 Duality

The *dual* of a matroid $M = (S, \mathcal{B})$ (not to be confused with the dual of an LP) is a matroid $M^{\mathrm{D}} = (S, \mathcal{B}^{\mathrm{D}})$ in which $B' \in \mathcal{B}^{\mathrm{D}}$ if and only if $S \setminus B' \in \mathcal{B}$. A base in the dual is the complement of a base in the original. The dual of the dual is, of course, the original matroid. In the case of a planar graph, the dual corresponds to the usual definition. See Fig. 1. Some additional facts about the dual (proofs omitted):

1. A circuit C in M is a *co-circuit* in M^{D} and vice-versa. In the case of graphs, co-circuits are cutsets. For example, h, i and b, d, g, e are cycles in G^{D} in Fig. 1 and they are cutsets in G. Conversely, a, c, h, i, f is a cycle in G and a cutset in G^{D}.

2. Contraction in the dual corresponds to deletion in the original and vice versa. Contracting an edge in the dual of a planar graph has the effect of coalescing the adjoining faces, i.e., deleting the edge in the original.

3. The dual of a graphic matroid is also graphic if and only if the underlying graph is planar.

These facts have important algorithmic consequences.

First, another theorem.

Theorem 3 *If C is a circuit of a matroid $M = (S, \mathcal{B})$ and D is a circuit in M^D, also called a co-circuit, then $|C \cap D| \neq 1$.*

Using Theorem 3 you can show that if x is a maximum cost element of a circuit, then there exists a minimum cost base that does not include x. So another greedy minimum spanning tree algorithm involves repeatedly finding maximum cost edges in circuits and removing them until the result is a tree. An efficient algorithm based on this idea is a randomized one by Karger et al. [23]. If we "dualize" the circuit element removal algorithm – deletion in the original corresponds to contraction in the dual – we get a version of Greedy-Min-Cost that finds the maximum weight complement of a spanning tree/forest.

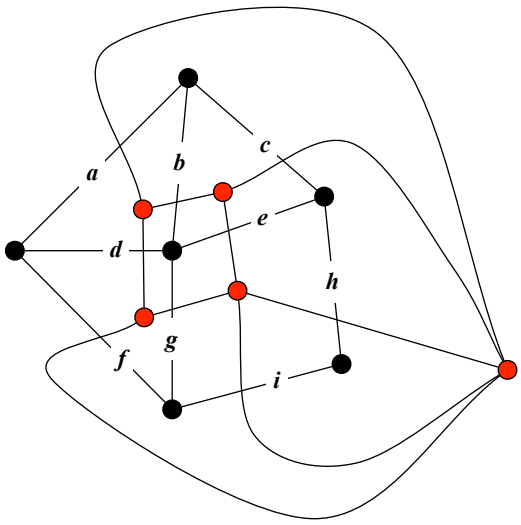
Even more important algorithmically is that the circuit removal algorithm, when applied to the dual of a graphic matroid yields an algorithm that repeatedly chooses a minimum cost edge of a cut set, the underlying framework of most of the known minimum spanning tree algorithms.

You might think this works only for planar graphs. Not true: co-circuits in a graphic matroid are still cutsets of the graph. The only issue is that the dual may not be graphic.

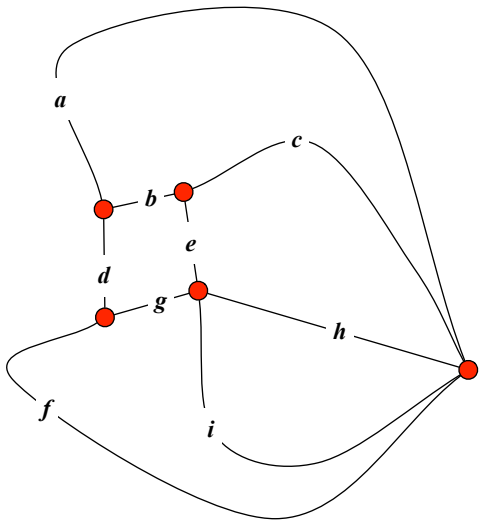
3.2 Minors and closure properties

A *minor* of a matroid M is any matroid that can be derived from M by a sequence of deletions and contractions. Both graphic and scheduling matroids are closed under minors.

Duality is a different story as we have already seen for graphic matroids. The class of *binary matroids*, matroids whose independent sets are independent vectors of 0's and 1's, is closed under minors and duality and includes both graphic matroids and their duals, co-graphic matroids. A binary matrix corresponding to a graph is the vertex/edge incidence matrix. To get closure under dual, invoke a standard representation with respect to a base $B = \{b_1, \ldots, b_r\}$:



(a) A planar graph $G = (V, E)$ and the graph induced by putting a vertex in each face and connecting adjacent faces.



(b) The dual $G^{\mathrm{D}} = (V^{\mathrm{D}}, E)$ of G. Note that G^{D} has the same edges as G, but V^{D} is the set of faces of G.

Figure 1: A planar graph and its dual. The complement of a spanning tree in the original graph is a spanning tree in the dual and vice-versa.

	a	b	c	g	h	d	e	f	i
a	1	0	0	0	0	1	0	1	0
b	0	1	0	0	0	1	1	1	1
c	0	0	1	0	0	0	1	0	1
g	0	0	0	1	0	0	0	1	1
h	0	0	0	0	1	0	0	0	1

(a) Standard representation with respect to $B = \{a, b, c, g, h\}$.

	d	e	f	i	a	b	c	g	h
d	1	0	0	0	1	1	0	0	0
e	0	1	0	0	0	1	1	0	0
f	0	0	1	0	1	1	0	1	0
i	0	0	0	1	0	1	1	1	1

(b) Representation of the dual with respect to $E \setminus B$.

Figure 2: Binary representations of the matroid corresponding to graph G in Fig. 1 and its dual.

$b_1, \ldots, b_r$	$e_1, \ldots, e_r$
I_r	A

The representation for the dual is

$e_1, \ldots, e_r$	$b_1, \ldots, b_r$
I_q	A^{T}

For the example in Fig. 1 the two matrices with respect to base $\{a, b, c, g, h\}$ are shown in Fig. 2. These matrices are no longer vertex/edge incidence matrices but they do have an interesting property: if $x \in E - B$ then the 1's in the column for x correspond to $C(x, B)$, i.e., the endpoints of the edges in the fundamental circuit each appear exactly twice and therefore cancel out. Contraction in a binary matroid is equivalent to deleting a column and adding it to each of the others.

An obvious generalization of a binary matroid is a *linear matroid*, where independence corresponds to independence of a set of vectors over any field. Linear matroids were the original concrete model on which matroid theory was built [34].

The dual of a scheduling matroid $M = (S, \mathcal{B})$ is also a scheduling matroid $M^{\mathrm{D}} = (S, \mathcal{B}^{\mathrm{D}})$. First assume a standard representation in which the maximum deadline is $n = \rho(S)$. Note that if there are k tasks with deadline n, the latest deadline, and the second latest deadline is $n - \delta$ then, in the original matroid, at least δ of the k tasks must be scheduled. So in the dual, at most $k - \delta$ of these tasks should be scheduled. This observation leads to the following recursive definition of the dual of $M = (S, \mathcal{B})$.

- if $n = 0$ then $\mathcal{B}^{\mathrm{D}} = \{S\}$, i.e., all tasks have deadline $m = |S|$ and S is the unique base
- otherwise let $n - \delta$ be the latest deadline of a task with deadline $< n$ (or n if there are no other tasks) and let $S_n = \{i \mid d_i = n\}$ and let $k = |S_n|$
- then M^{D} includes the tasks of S_n, each with deadline $k - \delta$, plus the dual of $M = (S', \mathcal{B}')$, where $S' = S \setminus S_n$ and $\mathcal{B}' = \{B \cap S' \mid B \in \mathcal{B}\}$, with $k - \delta$ added to each deadline

This process corresponds to the scheduling algorithm presented earlier, which we can now interpret as finding a minimum profit base in the dual by repeatedly deleting a maximum profit task from a circuit.

Fig. 3 illustrates the process of constructing the dual on an example. The three tasks g, h, i have deadline n and $\delta = 2$: we give them deadline 1. Now we find the dual of the matroid with tasks a–f: $n = 3$, $\delta = 1$, $S_n = \{e, f\}$; deadline is 1, but we add 1. Deadlines for the remaining tasks are determined similarly.

A more interesting example is shown in Fig. 4

a b	c d	e f		g h i
1	2	3	4	5

The matroid M. Here, any base must have, e.g., at most one of a or b, at most two of a, b, c, d, at least two of g, h, i and at least one of e, f if not all of g, h, i are included.

g h i	e f	c d	a b
1	2	3	4

The matroid M^{D}. Included in any base are at least one of a, b, at least two of a, b, c, d, at most one of g, h, i and at most one of e, f if one of g, h, i is included.

Figure 3: A scheduling matroid and its dual. Tasks are shown in columns representing their deadlines.

		b c d e
a		
1	2	3

The matroid M. Any base must have at least two of b, c, d, e and may or may not have a.

	b c d e
	a
1	2

The matroid M^{D}. Because $\delta = 2$ and $k = 4$ the tasks b, c, d, e are given a deadline of 2. In the reduced instance $n = \delta = k = 1$ (only one task, a, with deadline 1). So the deadline of a in the dual is 0, which is expanded to 2.

Figure 4: Dual of a different scheduling matroid.

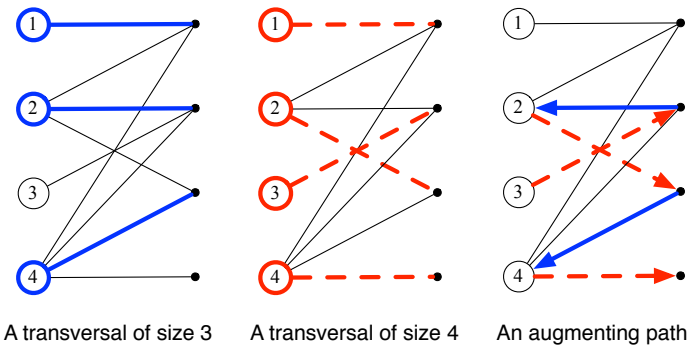


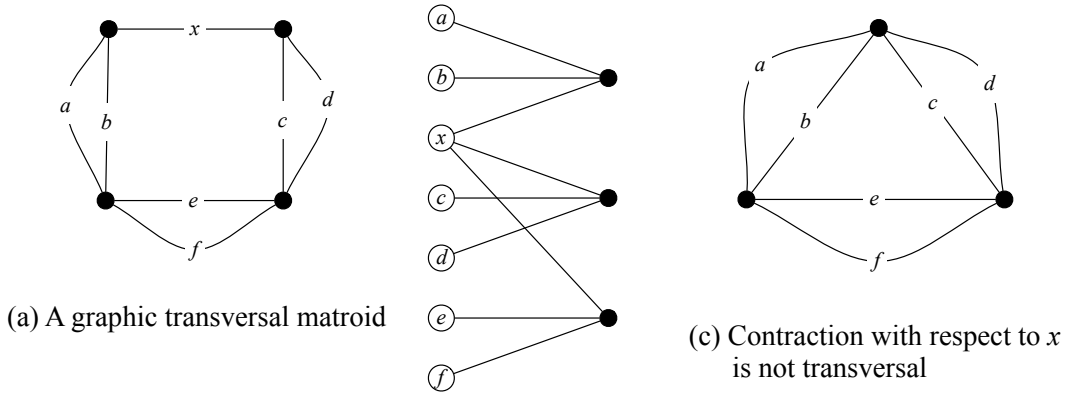
Figure 5: How axiom (I3) is satisfied in a transversal matroid.

A generalization of the class of scheduling matroids is *transversal matroids*. Let $G = (V_1, V_2, E)$ be a bipartite graph. Then $M = (V_1, \mathcal{I})$, where $X \subseteq V_1 \in \mathcal{I}$ if there exists a matching that covers the vertices in X, is a transversal matroid. Clearly (I1) and (I2) are satisfied. For (I3) let X and Y be in $\mathcal{I}$ with $|X| = |Y| + 1$ and let M_X and M_Y be the corresponding matchings. M_Y must leave some vertex x of X uncovered. Observe that $M_X \oplus M_Y$, the symmetric difference of M_X and M_Y, is the disjoint union of paths and cycles. One of the paths P must lead from x to an uncovered vertex $z \in V_2$. Then $(M_X \setminus P) \cup (M_Y \oplus P)$ is a matching that covers all of Y in addition to x. See Fig. 5

Side note: P is usually referred to as an augmenting path. The bipartite matching problem associated with a transversal matroid can also be formulated as a max flow problem with a source s, an edge with unit capacity directed into each $v \in V_1$ and a sink t with a unit capacity edge wt for each $w \in V_2$.

The augmenting path argument can be used for the more general class of *matching matroids*, based on non-bipartite graph matching. The corresponding flow problem is more complex. Surprisingly, it is also true that every matching matroid is also a transversal matroid – the construction is not at all obvious – so matching matroids are not really more general. A scheduling matroid is a special case of a *convex transversal matroid*, one in which there exists a permutation π of V_2 such that, for every $x \in V_1$ the set $\{y \mid xy \in E\}$ is contiguous in π.

Unfortunately neither transversal or matching matroids are closed under minors. Consider the example in Fig. 6. The graphic matroid in Fig. 6(a) is also a transversal matroid, as illustrated in Fig. 6(b), but when edge x is contracted – Fig. 6(c) – the result is not transversal. A circuit in a transversal matroid consists of k elements that have $k-1$ neighbors among them (but not every set with this property is a circuit). In Fig. 6(c), $\{a, b\}$, $\{c, d\}$ and $\{e, f\}$ are circuits; so a and b share a neighbor, as do c and d and e and f. No two of these neighbors can be the same: otherwise, for example, $\{a, c\}$ would also be a circuit.

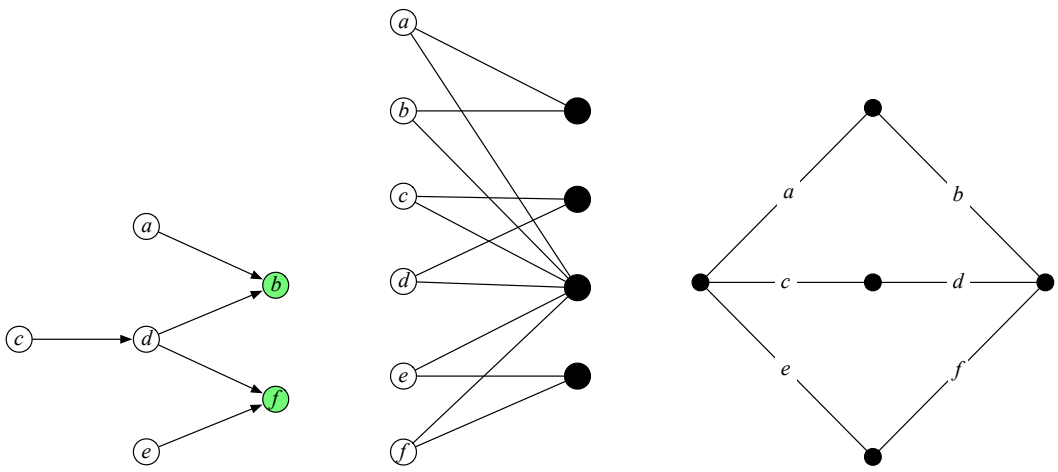


(a) A graphic transversal matroid

(b) Its representation as a transversal matroid

(c) Contraction with respect to x is not transversal

Figure 6: A transversal matroid whose contraction with respect to a given element is not transversal.



(a) The matroid M of Fig. 6(c) as a strict gammoid.

(b) The dual of the matroid M shown as a transversal matroid.

(b) The dual of the matroid M shown as a graphic matroid.

Figure 7: A strict gammoid and its dual.

The smallest class that includes transversal matroids and is closed under both minors and duals is the class of *gammoids*, defined as follows.

- Let $G = (V, E)$ be a directed graph and $W \subseteq V$ a fixed set of vertices.
- There is a *linking* of $U \subseteq V$ into W if there is a mapping $f : U \mapsto W$ such that there is a set of disjoint paths, each path leading from $u \in U$ to $f(u) \in W$.
- For any fixed W the set of U such that there exists a linking from U into W form the independent sets of a matroid over V. Such a matroid is called a *strict gammoid*.
- A gammoid is obtained by restricting U to some subset of V.

Fig. 7(a) illustrates the strict gammoid corresponding to the graphic matroid in Fig. 6(c). A matroid is a strict gammoid if and only if it is the dual of a transversal matroid. Proofs of this are given in the textbooks by Aigner [2] and Welsh [33]. Figures 7(b) and (c) show the dual of the strict gammoid in Fig. 7(a) as a transversal and graphic matroid respectively. It is easy to see that a transversal matroid M is a gammoid – let W be the V_2 in the matching representation of M and restrict the resulting strict gammoid to the vertices of V_1.

Gammoids, like binary matroids, are linear matroids. There are many ways to define a hierarchy on gammoids. Here is one. The simplest closed subclass is that of *uniform matroids*: $U_{n,m}$ is a matroid on m elements in which every subset of size n is a base. The sum of a set of uniform matroids is called a *partition matroid*. In general, if $M_1 = (S_1, \mathcal{B}_1)$ and $M_2 = (S_2, \mathcal{B}_2)$ are two matroids with $S_1 \cap S_2 = \emptyset$, then their sum, $M_1 + M_2$, is a matroid $M = (S_1 \cup S_2, \mathcal{B})$ where $\mathcal{B} = \{B_1 \cup B_2 \mid B_1 \in \mathcal{B}_1 \text{ and } B_2 \in \mathcal{B}_2\}$. A min-cost or max-profit base in a partition matroid can be found using a linear-time median algorithm.

Uniform matroids are also scheduling matroids: $U_{n,m}$ is equivalent to a scheduling matroid with m tasks each having deadline n. Partition and scheduling matroids therefore agree on uniform matroids but then diverge on rank-two matroids with four elements. Let $[m_1, m_2]$ denote a (rank-two) scheduling matroid with m_i tasks having deadline i. A four element rank-two scheduling matroid has either six bases ($[0, 4]$ or $[1, 3]$), five bases ($[2, 2]$ – the only non-base is the set of two deadline 1 tasks) or three bases ($[3, 1]$). Rank-two partition matroids have six bases ($U_{2,4}$), four bases ($U_{1,2} + U_{1,2}$) or three bases ($U_{1,3} + U_{1,1}$). So the scheduling matroid $[2, 2]$ cannot be a partition matroid and the partition matroid $U_{1,2}+U_{1,2}$ cannot be a scheduling matroid. The only non-uniform partition matroid that is also a scheduling matroid is $U_{1,2} + U_{1,1}$

A matroid M is binary if and only if it does not have $U_{2,4}$ as a minor. The "only if" part is easy: $U_{2,4}$ requires exactly two rows and four columns; in a binary matroid, one of those columns would have to be the 0 vector. A gammoid is binary if and only if it does not have K_4 as a minor, i.e., if it is the graphic matroid of a series-parallel graph.

Fig. 8 shows the relationships among the matroid classes discussed above.

4 More matroid algorithmics

Beyond finding the optimum base of a matroid there are additional, more challenging, problems, two of which are addressed here.

4.1 Matroid intersection

An instance of *matroid intersection* consists of two matroids over the same ground set $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$. The objective is to find $X \subseteq S$ such that $X \in \mathcal{I}_1 \cap \mathcal{I}_2$, i.e., X is independent in both matroids. Even a maximum cardinality X may not be trivial to find. An example of matroid intersection is bipartite matching. In contrast to the transversal matroid, the elements are the *edges* of a bipartite graph $G = (V_1, V_2, E)$ and a set of edges must be independent in two partition matroids:

$$M_1 = \sum_{v \in V_1} U_{1,\deg(v)} \text{ and } M_1 = \sum_{v \in V_2} U_{1,\deg(v)}$$

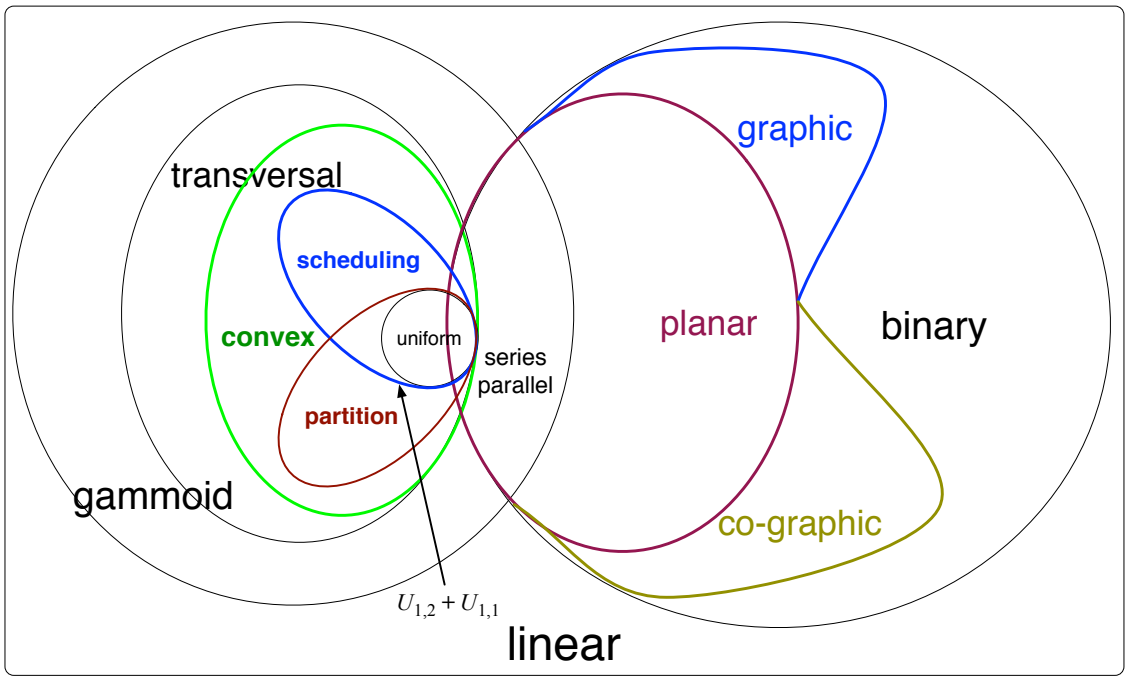


Figure 8: A (partial) hierarchy of matroid classes (not to scale).

Here, each $U_{1,\deg(v)}$ contains edges incident on v, i.e., at most one such edge can be included in an independent set.

Fig. 9 shows how given *maximal* matching can be extended to a *maximum* matching via discovery of an augmenting path. Taking the symmetric difference along the path $fdehi$ yields matching $\{a, e, f, i\}$. In general, a sequence of $O(\sqrt{n})$ augmenting paths leads from an empty matching to a maximum matching, as shown by Hopcroft and Karp [21].

The weighted bipartite matching problem, i.e., find a matching of maximum weight given weights on the edges, can be formulated as a linear program. Let $x_{ij} = 1$ if edge ij with weight w_{ij} is to be included in the matching, 0 otherwise. Here $i \in V_1$ and $j \in V_2$. The objective is to maximize $\sum_{ij \in E} w_{ij} x_{ij}$ subject to $\sum_{ij} x_{ij} \leq 1$ for every $i \in V_1$ and $\sum_{ij} x_{ij} \leq 1$ for every $j \in V_2$. The constraints ensure that each vertex has at most one incident edge in the matching. Early methods for solving weighted bipartite matching, sometimes known as Hungarian methods (in recognition of the König-Egervary Theorem equating maximum matching and minimum vertext cover) used the dual problem. Let y_i be the dual variable corresponding to the constraint for $i \in V_1$ and z_j the variable for the constraint for $j \in V_2$. Then the objective is to minimize $\sum_{i \in V_1} y_i + \sum_{i \in V_2} z_j$ subject to $y_i + z_j \geq w_{ij}$ for every $ij \in E$.

The most well-known Hungarian method, as described by Lawler [25], does a sequence of breadth-first searches like the cardinality matching algorithm. The algorithm makes use of the fact that an optimum primal/dual solution is found when the following *complementary slackness* conditions hold:

1. if $x_{ij} > 0$ then $y_i + z_j = w_{ij}$
2. if $y_i > 0$ then $\sum_{ij} x_{ij} = 1$ (the sum is over edges incident on i)
3. if $z_j > 0$ then $\sum_{ij} x_{ij} = 1$ (the sum is over edges incident on j)

The conditions say that whenever a dual variable is nonzero, the corresponding constraint is satisfied

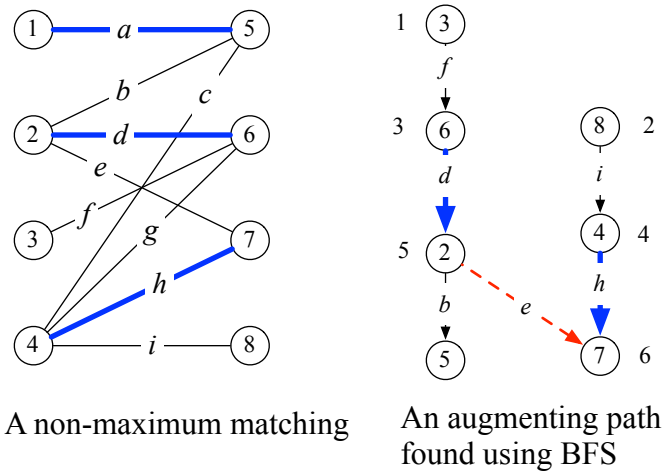


Figure 9: A breadth-first search finds an augmenting path with respect to a maximal matching.

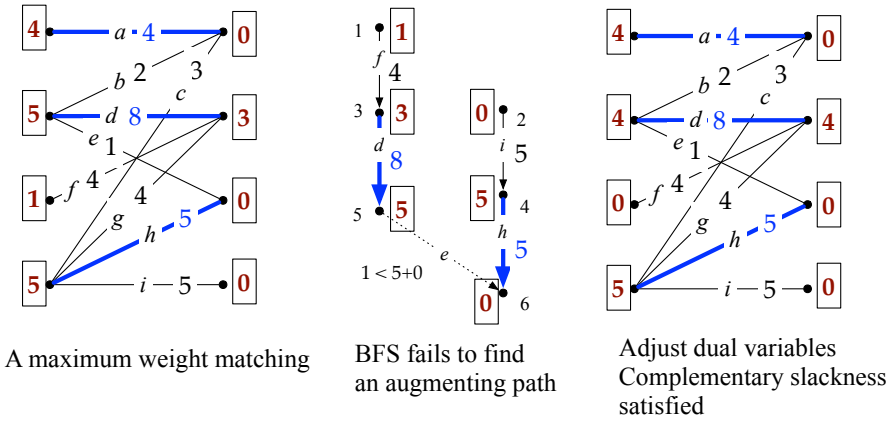
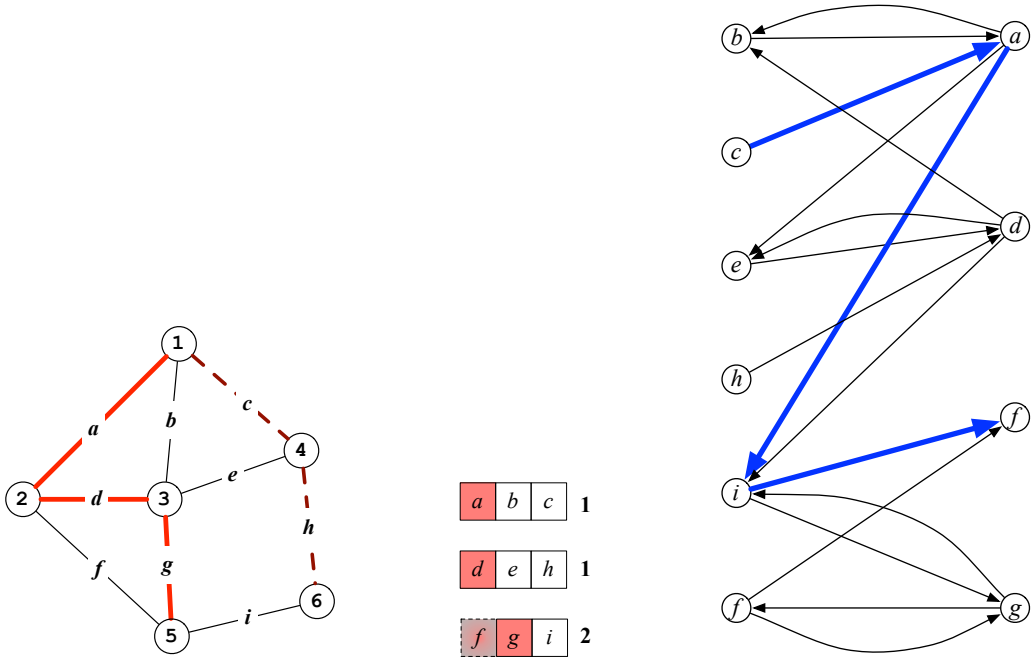


Figure 10: The final search and subsequent dual-variable adjustment in a Hungarian method for weighted bipartite matching.



(a) Graphic matroid M_1 (b) Partition matroid M_2 (c) Dependence graph

Figure 11: The dependence graph for a matroid intersection and an augmenting path.

with equality. The searches for augmenting paths maintain primal and dual feasibility and (1) and (3) by starting with all $x_{ij} = z_j = 0$ and $y_i = \max_{ij} w_{ij}$ – so (2) is not satisified until the end. An edge ij can participate in a search only if it is *tight*, i.e., $w_{ij} = x_i + y_j$. Thus ((1) is maintained. Using auxiliary variables during the search, the algorithm, when it gets stuck, adjusts dual variables; it decreases all of the y_i by a fixed amount δ, based on current conditions, and increases some, but not all of the z_j by δ. Fig. 10 shows the final adjustment, after the last, unsuccessful search. The edge e, which would have been used in an augmenting path in the cardinality problem, cannot be used here because it is not tight. In this example, the first adjustment reduces all y_i from 8 to 5 and increases z_j to 3 only for the other endpoint of edge d. This makes edges h and i tight while preserving both primal and dual feasibility.

The process of augmenting and adjusting dual variables is reminiscent of many network flow algorithms. This is no accident – the weighted bipartite matching problem, also known as the *assignment problem*, is often formulated as a minimum cost flow problem.

Matroid intersection problems involving more general matroids can also be solved using variations of the augmenting paths idea. Algorithms involving intersections of graphic or scheduling matroids with partition matroids are discussed by Gabow and Tarjan [17]. The basic idea is to find a mimimum cost base (if minimum cost is the objective) followed by a sequence of swaps until the constraints of the partition matroid are satisfied, if possible.

Tong, Lawler and Vazirani [31] were able to show that matroid intersection problems on gammoids reduce to bipartite matching (in the cardinality case) and weighted non-bipartite matching, discussed below, in the weighted case. On the other side of the matroid hierarchy, more sophistication is required in the process of finding augmenting paths. As will become evident in the later discussion of matroid parity, the addition or removal of an element from either of the two matroids impacts the relationships among the other elements in nontrivial ways.

The good news is that there is a static auxiliary graph, the *dependence graph*, that captures all swaps

that can take place in each matroid. Let X be the current independent set and let B_1 and B_2 be bases in M_1 and M_2, respectively so that $X \subseteq B_i$ for $i = 1, 2$. Construct the (directed) dependence graph $G_X = (S_1, S_2, A)$ with respect to X as follows.

- create a bipartition using two copies of each matroid element, so $S_1 = S_2 = S$
- if $y \in C(x, B_1)$ add arc xy to A
- if $x \in C(y, B_2)$ add arc yx to A

The arc xy corresponding to $y \in C(x, B_1)$ captures the idea that, if y is *removed* from B_1, x can replace it. The arc yx corresponding to $x \in C(y, B_2)$ captures the idea that, if y is added to B_2, x can be removed from it. A sequence of such swaps yields a path in G_X; call it an augmenting path if it begins with an element $y \notin X$ and ends with an element $z \notin X$. As already noted, the path may not lead to a new independent set because any swap will affect the identity of the cycles. However, Krogdahl [24] showed that an augmenting path P in the dependence graph leads to a larger independent set if P admits no shortcuts.

Fig. 11 shows the dependence graph for a matroid intersection problem involving the intersection of a graphic matroid M_1 and a partition matroid M_2. The latter is $U_{1,3} + U_{1,3} + U_{2,3}$ with sets $\{a, b, c\}$, $\{d, e, h\}$ and $\{f, g, i\}$, respectively. The current independent set $X = \{a, d, g\}$ is maximal in both M_1 and M_2. To turn this into a base B_1 of M_1 we add edges c and h. To get a base B_2 in M_2 we add element f to bring the number in the partition containing g to 2. The added elements are parallel to existing ones that can replace them if an augmenting path is found. The augmenting path shown in Fig. 11(c) indicates that adding c to B_2 forces the removal of a. In M_1, elements b, e and i have a in their fundamental circuits. The augmenting path suggests adding i and then removing f from B_2. Since the latter is not in X, the sequence of swaps leads to $\{c, d, g, i\}$, a larger set independent in both M_1 and M_2.

For general matroids (not necessarily linear) an *oracle algorithm* is used in conjunction with this augmenting path idea. An oracle algorithm treats the matroid as a black box which can be queried: give it a set X and it returns the rank $\rho(X)$. Lawler [25] describes an oracle algorithm for cardinality matroid intersection that runs in time $O(mn^2\rho)$, where ρ is the time for an oracle query. This was improved to $O(mn^2 \lg n)$ for linear matroids by Cunningham [6]. For graphic and cographic matroids, particularly planar graphs, clever data structures can be used to keep track of cycles and the bounds are further reduced to $O(mn^{1/2})$ and $O(n^{3/2} \lg n)$ for graphic/co-graphic and planar graphs, respectively – see Gabow and Stallmann [15].

In constrast to the matching problem, formulating a linear program for the weighted case is a challenge. Instead of a simple constraint at each vertex there has to be a constraint of the form $\sum_{i \in C} x_i \leq |C| - 1$ for each circuit C and there are exponentially many circuits. The trick is to add these constraints on an as needed basis. Various minimum spanning tree algorithms (simple optimization, not matroid intersection) do this implicitly – see, e.g., Ahuja et al. [1]. For general matroids (not necessarily linear) an *oracle algorithm* is used in conjunction with the Hungarian method as described by Lawler [25] – runtime is $O(mn^3 + mn^2\rho)$, where ρ is the time for an oracle query. For linear matroids, Frank [8] improved this to $O(m\mathsf{M}(n))$, where $\mathsf{M}(n)$ is the time to do matrix multiplication (or, in this case, solving a system of linear equations). This can probably be improved for graphic and co-graphic matroids and especially planar graphs but the results are not in yet (as far as I know).

4.2 Matroid parity

The *matroid parity problem* (sometimes called *matroid matching*) is formulated as follows: Input is a matroid $M = (S, \mathcal{I})$ and a partition of S into pairs of elements, i.e., each element $x \in S$ has a unique mate $x' \in S$. The objective is to find the maximum cardinality (or weight if the pairs are weighted) $X \in \mathcal{I}$ such that, for each pair x, x', the element $x \in X$ if and only its mate $x' \in X$, i.e., the pairs must stay together. A special case of matroid parity is nonbipartite graph matching. Here M is a partition matroid composed from graph $G = (V, E)$ as $\sum_{v \in V} U_{1,\deg(v)}$. Think of each vertex

being a uniform matroid over the edges incident on it, but only using the half of each edge belonging to the vertex, independent of the other endpoint of the edge. Now an edge is a pair representing its two endpoints, each member of the pair participating in one uniform matroid.

Matroid parity generalizes all the problems discussed so far, as well as a large collection of network flow problems. In particular it generalizes both nonbipartite matching (and, by extension, bipartite matching) and matroid intersection. The latter is easy to see. Consider a matroid intersection problem defined by $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$. Let $M = M_1 + M_2$, i.e., create two copies of S, one for each matroid, and create a pair from the two copies of each element.

Nonbipartite matching also has an augmenting path theorem: a matching is maximum if and only if it admits no augmenting path, but the situation is more complicated algorithmically than for bipartite matching. Fig. 12 illustrates how two exploratory paths in a search may meet and *not* yield an augmenting path even though one exists. In bipartite matching, when two paths meet, they either form an augmenting path or an even length cycle. In the latter case, either path through the cycle can be used as part of an augmenting path if one exists.

The problem in nonbipartite matching is that an "exit", such as the one involving edge j in Fig. 12 may be missed because the alternation between matched and unmatched edges is different on opposite sides of the cycle. Edmonds [7] figured out that the odd cycle, which he referred to as a blossom, could simply be contracted into a single vertex, after which the search could proceed. Blossoms can also be incorporated into a linear programming formulation of the problem. In an odd cycle of length k, at most $\lfloor k/2 \rfloor$ edges can be included. Fig. 12 shows the full LP formulation with the dual variable z_β corresponding to the blossom β. Blossoms can be nested arbitrarily, but the search still succeeds if there is, indeed, an augmenting path. Furthermore, the LP formulation can be used to obtain an algorithm for weighted nonbipartite matching. Edmond's $O(n^4)$ algorithm was implemented by Gabow [10] to run in time $O(n^3)$. This was further reduced to $O(mn^{1/2})$ by Micali and Vazirani [28], matching (hah) the bound for the bipartite case.

Matroid parity, like matroid intersection, can be reduced to matching when the underlying matroid is a gammoid [31]. This works for both the cardinality and weighted cases.

The other side of the matroid hierarchy is much more interesting. Gary and Johnson [20] list *spanning tree parity*, matroid parity in graphic and/or co-graphic matroids, as an open problem. That same year[2] Lovasz [26] showed that the cardinality parity problem for linear matroids could be solved in polynomial time, albeit with a high degree polynomial. Estimates range from $O(m^2n^5)$ for (co-)graphic matroids[3] to $O(n^{10})$ for linear matroids [27]. Furthermore, the algorithm does not resemble any algorithms for the problems that matroid parity generalizes. Lovasz and, independently, Jensen and Korte [22], also gave an exponential lower bound for any oracle algorithm solving the matroid parity problem; contrast this with matroid intersection.

As with all the special cases of matroid parity, there is an augmenting path theorem. The primary issue is how to ensure that an augmenting path, if one exists, may be found using a search.

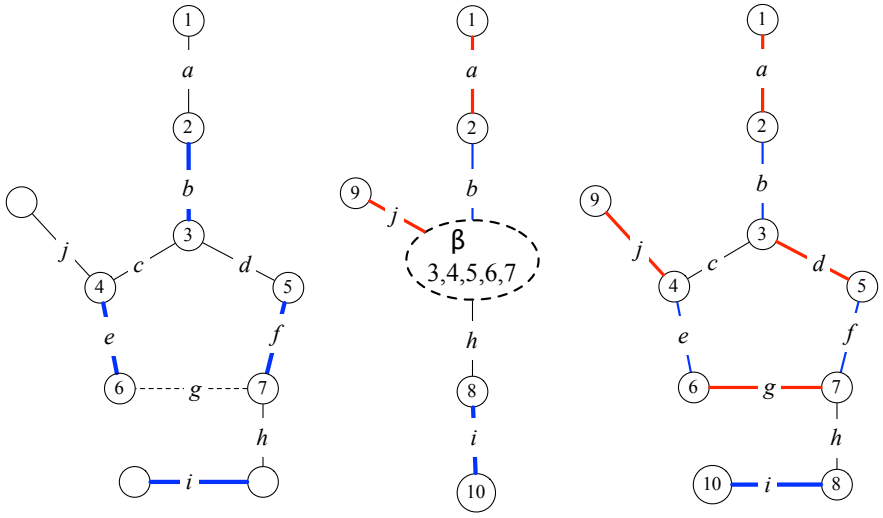
To get an idea of the pitfalls when solving the graphic matroid parity problem consider the example in Fig. 13. A standard breadth-first labeling starting at a determines that either c or e can be added when b' is removed. Then c' forces removal of d and e' forces removal of d'. How the search proceeds depends on which of c or e is encountered first. But regardless, the option of removing g, g' after adding *both* c and e is missed.

There is no obvious shrinking operation here. Instead, the solution is to create a *transform* $T(b', c, e)$ that captures all feasible exits from the blossom. In a standard representation of the binary matroid, let $[z_{b'} z_g z_h]$ be the projection onto the subspace induced by b', g, h. Then $c = [101]$ (its fundamental cycle includes b' and h) and $e = [111]$ (the cycle includes all three). $T(b', c, e) = [101] \oplus [111] = [010]$,[4]

[2] Actually it was a year earlier, but not well known outside of Hungary.

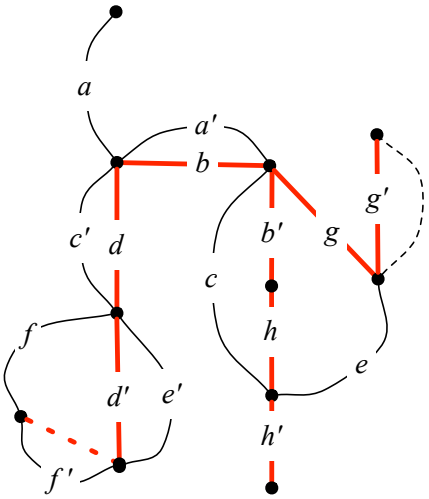
[3] Based on a Lisp implementation by Stallmann, which may still be available in electronic form if anyone really, really wants it.

[4] In a more general linear matroid, the two vectors would each be multiplied by a different scalar related to b'; here that scalar is 1 for both of them.

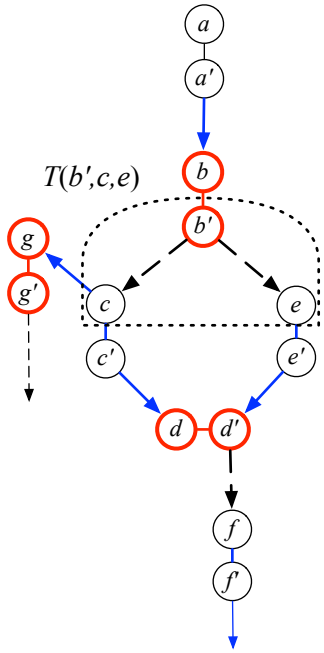


primal		**dual**	
$\max\ x_a + x_b + x_c + x_d + x_e + x_f + x_g + x_h + x_i + x_j$		$\min\ y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} + 2z_\beta$	
$x_a \le 1$	(vertex 1)	$y_1 + y_2 \ge 1$	(edge a)
$x_a + x_b \le 1$	(vertex 2)	$y_2 + y_3 \ge 1$	(edge b)
$x_b + x_c + x_d \le 1$	(vertex 3)	$y_3 + y_4 + z_\beta \ge 1$	(edge c)
$x_c + x_e + x_j \le 1$	(vertex 4)	$y_3 + y_5 + z_\beta \ge 1$	(edge d)
$x_d + x_f \le 1$	(vertex 5)	$y_5 + y_7 + z_\beta \ge 1$	(edge f)
$x_e + x_g \le 1$	(vertex 6)	$y_4 + y_6 + z_\beta \ge 1$	(edge e)
$x_f + x_g + x_h \le 1$	(vertex 7)	$y_6 + y_7 + z_\beta \ge 1$	(edge g)
$x_h + x_i \le 1$	(vertex 8)	$y_7 + y_8 \ge 1$	(edge h)
$x_i \le 1$	(vertex 10)	$y_8 + y_{10} \ge 1$	(edge i)
$x_j \le 1$	(vertex 9)	$y_4 + y_9 \ge 1$	(edge j)
$x_c + x_d + x_e + x_f + x_g \le 2$	(blossom)		
solution		**solution**	
$x_a = x_d = x_g = x_j = 1$		$y_8 = y_9 = z_\beta = 1$	
all others 0			

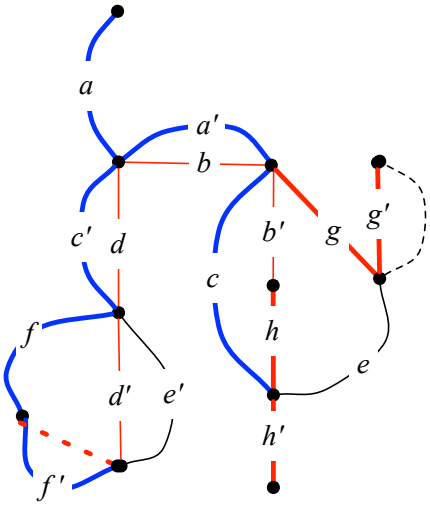
Figure 12: An example of a search for an augmenting path eith respect to a matching in a nonbipartite graph. Also shown are the primal and dual LP formulations and the values of variables in the optimumn solution.



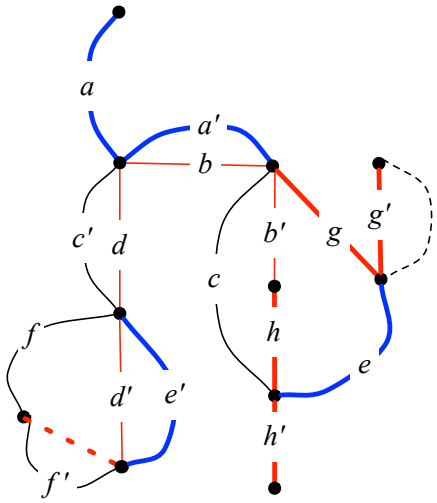
(a) an instance of graphic matroid parity; elements with thick, straight red lines are included in the current independent set



(b) the search for an augmenting path encounters a classic blossom situation



(c) if the pair c, c' is added first, f, f' can be added, but the pair g, g' is ignored completely



(d) if the pair e, e' is added first, adding f, f' is not possible and the pair g, g' is "blocked"

Figure 13: An example illustrating what a blossom would look like in the context of an augmenting path algorithm for graphic matroid parity.

matroid class	optimization (greedy)	cardinality intersection	cardinality parity	weighted intersection	weighted parity
partition	$O(m)$	$O(\mathsf{BM}(m,n))$ [31]	$O(\mathsf{GM}(m,n))$ [11, 32]	$O(\mathsf{WB}(m,n))$ [11]	$O(\mathsf{WG}(m,n))$ [11, 32]
scheduling	$O(m + n \lg n)$ [17]	$O(\mathsf{BM}(m,n))$[31]	$O(\mathsf{GM}(m,n))$ [31]	$O(\mathsf{WG}(m,n))$ [31]	$O(\mathsf{WG}(m,n))$ [31]
gammoid[a]	$O(\mathsf{BM}(m,n))$	$O(\mathsf{BM}(m,n))$ [31]	$O(\mathsf{GM}(m,n))$ [31]	$O(\mathsf{WG}(m,n))$ [31]	$O(\mathsf{WG}(m,n))$ [31]
planar	$O(n)$ [4]	$O(n^{3/2} \lg n)$ [15]	$O(mn \lg^6 n)$ [15]	$O(m\mathsf{M}(n))$ [8][b]	***open***
(co-)graphic	$O(m\alpha m, n)$ [3]	$O(mn^{1/2})$ [15][c]	$O(mn \lg^6 n)$ [15]	$O(m\mathsf{M}(n))$ [8]	***open***
linear	$O(\mathsf{M}(n))$	$O(mn^2 \lg n)$ [6][d]	$O(m\mathsf{M}(n))$ [16]	$O(m\mathsf{M}(n))$ [8]	***open***
general	$O(m \lg m + m\rho)$	$O(mn^2\rho)$ [25]	$\Omega(c^n)$ [26][e]	$O(mn^3 + mn^2\rho)$ [25]	$\Omega(c^n)$ [26]

[a]Time bounds for transversal/matching matroids extend to gammoids.

[b]There must be a better algorithm

[c]This bound applies if $m \in \Omega(n^{3/2} \lg n)$. If $m \in \Omega(n \lg n) \cap O(n^{3/2} \lg n)$ the bound is $O(m^{2/3} n \lg^{1/3} n)$. If $m \in O(n \lg n)$ it is $O(m^{1/3} n^{4/3} \lg^{2/3} n)$.

[d]Improved to $O(mn^{1.62})$ – don't ask – by Gabow and Xu [19]

[e]Here $c = 2 - \epsilon$ for $\epsilon \to 0$ as $n \to \infty$; uses an extension of the non-linear Vamos-Higgs matroid.

Table 1: Time bounds for various matroid related algorithms; some of these may be outdated. Here m is the number of elements and n the rank of the matroid. To indicate provenance of the time bounds, let $\mathsf{M}(n)$, $\mathsf{BM}(m,n)$, $\mathsf{WB}(m,n)$, $\mathsf{GM}(m,n)$ and $\mathsf{WG}(m,n)$ represent the bounds for matrix multiplication, bipartite matching, weighted bipartite matching, non-bipartite matching and weighted non-bipartite matching, respectively. The notation ρ, in context of time bounds for general matroids, refers to the time to query a rank or independence oracle.

which is exactly g as desired. It also captures the fact that h is not an option.

An algorithm that solves the matroid parity problem using blossom shrinking with transforms is reported by Gabow and Stallmann [16]; it has time bounds $O(m\mathsf{M}(n))$ for linear matroids and $O(mn^2)$ for (co-)graphic matroids. Using clever data structures the bound for (co-)graphic matroids can be improved to $O(mn \lg^6 n)$ [15].

Another line of attack on matroid parity involves a reduction to a sequence of matroid intersection problems, originally proposed by Orlin and VandeVate [30] and improved by Orlin [29]. Time bounds are $O(mn^4)$ and $O(m^3 n)$, respectively, for linear matroids.

In Table 1 gives the latest bounds (as far as I know) for algorithms solving a wide variety of matroid problems. Some of the entries depend on the following time bounds, also subject to improvement.

- matrix multiplication, $\mathsf{M}(n) \in O(n^{2.2\cdots})$, Coppersmith and Winograd [5].
- cardinality bipartite matching, $\mathsf{BM}(m,n) \in O(mn^{1/2})$, Hopcroft and Karp [21].
- cardinality nonbipartite matching, $\mathsf{GM}(m,n) \in O(mn^{1/2})$, Micali and Vazirani [28].
- weighted bipartite matching, $\mathsf{WB}(m,n) \in O(mn + n^2 \lg n)$, Fredman and Tarjan [9].
- weighted nonbipartite matching, $\mathsf{WG}(m,n) \in O(\min n^3, mn \lg n)$, Gabow, Galil and Spencer [13], improved to $O(mn + n^2 \lg n)$ by Gabow [12].

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Martin Aigner. *Combinatorial Theory*. Springer, 1979.

[3] Bernard Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *JACM*, pages 1028–1047, 2000.

[4] David Cheriton and Robert E. Tarjan. Finding minimum spanning trees. *SIAM J. Computing*, pages 724–742, 1976.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, pages 251–280, 1990.

[6] William H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Computing*, pages 948–957, 1986.

[7] Jack R. Edmonds. Paths, trees and flowers. *Canadian J. Math.*, pages 449–467, 1965.

[8] András Frank. A weighted matroid intersection algorithm. *J. Algorithms*, pages 328–336, 1981.

[9] Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *JACM*, pages 596–615, 1987.

[10] Harold N. Gabow. An efficient implementation of edmonds' algorithm for maximum matching on graphs. *J. ACM*, 23(2):221–234, April 1976.

[11] Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proc. 15th Ann. ACM Symp. on Theory of Computing*, 1983.

[12] Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. First Ann. Symp. on Discrete Algorithms*, 1990.

[13] Harold N. Gabow, Zvi Galil, and Thomas H. Spencer. Efficient implementation of graph algorithms using contraction. *JACM*, pages 540–572, 1989.

[14] Harold N. Gabow, Zvi Galil, Thomas H. Spencer, and Robert E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, pages 109–122, 1986.

[15] Harold N. Gabow and Matthias F. Stallmann. Efficient algorithms for graphic matroid intersection and parity. In *Automata, Languages and Programming*, number 194 in Lecture Notes in Computer Science, 1985.

[16] Harold N. Gabow and Matthias F. Stallmann. An augmenting path algorithm for linear matroid parity. *Combinatorica*, pages 123–150, 1986.

[17] Harold N. Gabow and Robert E. Tarjan. Efficient algorithms for simple matroid intersection problems. *Journal of Algorithms*, pages 80–131, 1984.

[18] Harold N. Gabow and Robert E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comp. and Syst. Science*, pages 209–221, 1985.

[19] Harold N. Gabow and Ying Xu. Efficient theoretic and practical algorithms for linear matroid intersection problems. *J. Comp. and Syst. Sci.*, pages 129–147, 1996.

[20] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[21] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Computing*, pages 225–231, 1973.

[22] Per M. Jensen and Bernhard Korte. Complexity of matroid property algorithms. *SIAM J. Computing*, pages 184–190, 1982.

[23] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *JACM*, pages 321–328, 1995.

[24] Stein Krogdahl. The dependence graph for bases in matroids. *Discrete Mathematics*, 19(1):47–59, 1977.

[25] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[26] Lázló Lovász. The matroid matching problem. In *Algebraic Methods in Graph Theory*, Colloquia Mathematica Societatis János Bolyai, pages 495–517, 1978.

[27] Lázló Lovász and M. D. Plummer. *Matching Theory.* North Holland, 1986.

[28] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matchings in general graphs. In *Proc. 21st Ann. Symp. on Foundations of Computer Science*, 1980.

[29] James B. Orlin. A fast, simpler algorithm for the matroid parity problem. In *Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 240–258, 2008.

[30] James B. Orlin and John H. VandeVate. Solving the matroid parity problem as a sequence of matroid intersection problems. *Mathematical Programming*, pages 81–106, 1990.

[31] Po Tong, Eugene Lawler, and Vijay V. Vazirani. Solving the weighted parity problem for gammoids by reduction to graphic matching. In *Progress in Combinatorial Optimization*, pages 363 – 374. Academic Press, 1984.

[32] Robert J. Urquhart. *Degree constrained subgraphs of linear graphs.* PhD thesis, University of Michigan, 1967.

[33] D.J.A Welsh. *Matroid Theory.* Academic Press, 1976.

[34] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, pages 509–533, 1935.

Acknowledgement

Thanks to Hal Gabow for getting me involved in matroid research and in memory of Gene Lawler, from whom I learned most of what I know about matroids.