CMPUT 675: Topics in Combinatorics and Optimization **Fall 2016**

Lecture 26 (November 14): Matroids

Lecturer: Zachary Friggstad *Scribe: Bradley Hauer*

26.1 Introduction to Matroids

A *matroid* is an abstract mathematical structure with applications to several areas of mathematics and computing science, including combinatorics and optimization. We will begin with the definition of a matroid, and, in the next section, will look at several examples.

Definition 1 *Let X be a finite set of items, and $\mathcal{I} \subseteq 2^X$ (where 2^X is the set of subsets of X). Then $M = (X, \mathcal{I})$ is a* **matroid** *if it satisfies all three of these properties:*

1. $\emptyset \in \mathcal{I}$
2. *If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$*
3. **(Exchange property)** *If $A, B \in \mathcal{I}$ and $|A| < |B|$, then $\exists\, z \in B - A$ such that $A \cup \{z\} \in \mathcal{I}$*

The set X is typically called the *ground set*, and the sets in $\mathcal{I}$ are called the *independent sets* of the matroid. In the next section, we will see some examples that should help to clarify these ideas.

26.2 Examples of Matroids

26.2.1 Graphic Matroid

Let $G = (V; E)$ be a graph. Let $\mathcal{I} = \{F \subseteq E \;:\; (V; F) \text{ has no cycles}\}$. Then $M = (E, \mathcal{I})$ is a matroid.

Consider the three "matroid axioms" we saw above: The first clearly holds, since an empty set of edges trivially contains no cycles. The second is also easy to see, as if a set of edges contains no cycles, certainly no subset of that set can have a cycle.

The third property is only slightly harder to prove. If $A, B \in \mathcal{I}$ and $|A| < |B|$, then $(V; A)$ has more connected components than $(V; B)$, so there is some $z \in B$ that connects two components of $(V; A)$. To see this, use the pigeonhole principle to see there are two nodes u, v in different components of $(V; A)$ that lie in the same component of $(V; B)$, so some edge z of a $u - v$ path in $(V; B)$ bridges different components of $(V; A)$. Thus, $(V; A \cup \{z\})$ still has no cycles, therefore $A \cup \{z\} \in \mathcal{I}$. So M is indeed a matroid.

26.2.2 Vector Matroid

Let $\mathcal{V}$ be a vector space, and let $X \subseteq \mathcal{V}$ be finite. Let $\mathcal{I} = \{Y \subseteq X \;:\; \text{the vectors in } Y \text{ are linearly independent}\}$. Them $M = (X, \mathcal{I})$ is a matroid.

For example, X could be the columns of a matrix with all its columns distinct (or we could modify the definition slightly to allow X to be a multiset, if we want to allow matrices with identical columns). Then $\mathcal{I}$ is simply the set of sets of linearly independent vectors.

26.2.3 Laminar Matroid

Let X be a finite set of items. Let $\mathcal{L}_X$ be a laminar family over X (see Assignment 3 for a definition). Also, for each $A \in \mathcal{L}_X$, let $b_A \in \mathbb{Z}$, $b_A \geq 0$. Finally, let $\mathcal{I} = \{Y \subseteq X \ : \ |Y \cap A| \leq b_A \ \forall A \in \mathcal{L}_X\}$. Them $M = (X, \mathcal{I})$ is a matroid.

The b_A values set limits on the number of items which may be chosen from each set in the laminar family $\mathcal{L}_X$. Any subset which respects these limits, containing at most b_A items from each $A \in \mathcal{L}_X$, is an independent set.

26.2.4 Transversal Matroid

Let X be a finite set of items, and $\mathcal{F} \subseteq 2^X$. Let $\mathcal{I} = \{Y \subseteq X \ : \ \exists \text{ injective } \mathcal{T} : Y \to \mathcal{F} \, s.t. \forall\, a \in Y,\, a \in \mathcal{T}(a)\}$. Them $M = (X, \mathcal{I})$ is a matroid.

Essentially, this gives a matroid structure to the notion of a matching; a subset $Y \subset X$ is an independent set if there exists a matching that saturates Y.

26.2.5 Gammoid

Let $G = (V; E)$ be a directed graph. Let $A, B \subseteq V$. Let $\mathcal{I} = \{A' \subseteq A \ : \ \exists \text{ vertex disjoint paths from } A' \text{ to } B\}$. Then $M = (A, \mathcal{I})$ is a matroid.

The definition of $\mathcal{I}$ requires that there exists a set of vertex-disjoint paths, each starting at a vertex in A' and ending at a vertex in B, such that there is a path starting at each element of A'. Note that there is an obvious relationship between gammoids and maximum flow problems.

26.2.6 Bond Matroid

Let $G = (V; E)$ be an undirected, connected graph. Let $\mathcal{I} = \{F \subseteq E \ : \ (V; E - F) \text{ is connected}\}$. Then $M = (E, \mathcal{I})$ is a matroid.

So, a set of edges is an independent set if removing the edges in that set does not disconnect the graph.

26.2.7 Algebraic Matroid

Let F, K be fields and $F \subseteq K$ (so K is an extension field of F). Let X be a finite subset of K, and $\mathcal{I} = \{Y \subseteq X \ : \ F[Y] \text{ has transcendence degree } |Y|\}$. Them $M = (X, \mathcal{I})$ is a matroid.

The terminology used here goes a bit beyond the scope of this course; it is given here simply as another example.

Independence Oracles

We briefly discuss how we will represent matroids for the purpose of computation, since the set of independent sets could be exponentially large. For this we will use the concept of an *independence oracle*:

Definition 2 *An* **independence oracle** *for $M = (X, \mathcal{I})$ is an algorithm which, given $Y \subseteq X$, decides, in polynomial time, if $Y \in \mathcal{I}$.*

The notion of polynomial time in the definition of an independence oracle can get a bit murky. For example, with a vector matroid we have to test for linear independence of vectors. The point is that whenever we actually use matroids we are dealing with a larger input (e.g. a graph or a matrix) and each query should run in time that is polynomial in the input. We don't dwell on this issue much longer, the algorithms we discuss will only invoke the independence oracle a polynomial number of times. So, given an efficient implementation of such an oracle, the overall algorithm will run in polynomial time.

So, if an algorithm needs to take a matroid $M = (X, \mathcal{I})$ as input, we will assume it takes the set X, and an independence oracle which can check membership in $\mathcal{I}$ and that this check can be done efficiently. We will say that the matroid M is *specified* by the independence oracle.

26.3 Bases

In this section, we will introduce the concept of a *base* of a matroid, introduce a related combinatorial problem, and give an algorithm for solving it.

Now we can move on to bases.

Definition 3 *A* **base** *of a matroid $M = (X, \mathcal{I})$ is a set $B \in \mathcal{I}$ such that if $B \subseteq B'$ and $B' \in \mathcal{I}$, then $B = B'$.*

In other words, a base is a maximal independent set. In a graphic matroid over a connected graph (where independent sets are forests), a base is a spanning tree. In a vector matroid (where independent sets are sets of linearly independent vectors), a base is a basis for the span of the set of vectors. In a bond matroid, a base is a set of edges the removal of which leaves a spanning tree (i.e. removing any other edge would disconnect the graph).

It is a basic property of spanning trees that any spanning tree has size $|V| - 1$; so, in a graphic matroid, all bases are of equal size. We will now show that this property holds for matroids in general.

Lemma 1 *Let B and B' be bases for matroid $M = (X, \mathcal{I})$. Then $|B| = |B'|$.*

Proof. If $|B| < |B'|$, by the exchange property (property 3 in the definition of a matroid), there exists some $z \in B' - B$ such that $B \cup \{z\} \in \mathcal{I}$, so B is not maximal, and so is not a base. So we cannot have $|B| < |B'|$. Similarly, by simply reversing B and B' in the above, we cannot have $|B| > |B'|$. Therefore $|B| = |B'|$. ■

So, by this lemma, given a connected graph $G = (V; E)$, all maximal sets of edges whose removal does not disconnect G are of equal size, since these sets are the bases of the bond matroid for that graph. This also shows that all spanning trees are of equal size, all bases of the span of a set of vectors are of equal size, and so on.

26.4 The Minimum Weight Base Problem

Just as bases generalize the notion of spanning trees in a graph, there is also a natural generalization of the minimum spanning tree problem, which, as we will see in this section, can be solved by a generalization of Kruskal's algorithm.

First, we define our problem:

Definition 4 *Given a matroid $M = (X, \mathcal{I})$, specified by an independence oracle, and a weight function $w : X \to \mathbb{R}$ (note that weights can be negative), the goal of the* **Minimum Weight Base *Problem*** *is to find a base $B \in \mathcal{I}$ of M of minimum weight (where the weight of a base is the sum of the weights of its elements).*

An algorithm for MINIMUM WEIGHT BASE is as follows:

Algorithm 1 MINIMUM WEIGHT BASE Algorithm

Input: A matroid $M = (X, \mathcal{I})$, specified by an independence oracle, with item weights $w(x) \in \mathbb{R}, x \in X$.
Output: A minimum weight base B of M.

```
$B \leftarrow \emptyset$
for each item $x \in X$ in increasing order of weight $w(x)$ do
  if $B \cup \{x\} \in \mathcal{I}$ then
    $B \leftarrow B \cup \{x\}$
  end if
end for
return $B$
```

As mentioned above, this is easily recognizable as a generalization of Kruskal's algorithm. We will now prove its correctness, first by proving that the returned set B is a base at all – it is clearly an independent set, since B is initially empty, and the algorithm always maintains the independent set property, but we need to prove it is a base – and then by proving it is of minimum cost.

Lemma 2 *The set B returned by Algorithm 1 is a base of M.*

Proof. Suppose B is not a base. Then $B \cup \{x\} \in \mathcal{I}$ for some $x \notin B$. Let B' be the value of B when the algorithm considered x. The algorithm never removes an item from B, so $B' \subseteq B$, therefore $B' \cup \{x\} \subseteq B \cup \{x\}$. We have $B \cup \{x\} \in \mathcal{I}$, therefore $B' \cup \{x\} \in \mathcal{I}$. This implies that the algorithm would have added x to B', so we should have $x \in B$, a contradiction. So B is a base. ■

Lemma 3 *The set B returned by Algorithm 1 is a minimum weight base of M.*

Proof. Let B^* be a minimum weight base. Order both $B = \{x_{i_1}, \dots, x_{i_t}\}$ and $B^* = \{x_{i_1^*}, \dots, x_{i_t^*}\}$ (recall two bases have the same size) in increasing order of weight. Let $B_k = \{x_{i_1}, \dots, x_{i_k}\}$ and $B_k^* = \{x_{i_1^*}, \dots, x_{i_k^*}\}$ for any $0 \le k \le t$ (where $B_0 = B_0^* = \emptyset$).

If $w(B) > w(B^*)$ then consider the least k with $w(x_{i_k}) > w(x_{i_k^*})$. Now, $|B_{k-1}| < |B_k^*|$ so there is some $x \in B_k^* - B_{k-1}$ with $B_{k-1} \cup \{x\} \in \mathcal{I}$. Note $w(x) \le w(x_{i_k^*}) < w(x_{i_k})$, so x was considered by the algorithm before x_{i_k}. But then the algorithm should have added x to B, a contradiction. Therefore $w(B) \le w(B^*)$. ■