Matroid Theory

1 Matroids and the Greedy Algorithm

Many combinatorial optimization problems concern choosing the best set from a family of possible sets. We can then define a **set system** as $(\mathcal{X}, \mathcal{I})$ where $\mathcal{X}$ is the **ground set** and $\mathcal{I} \subseteq 2^{\mathcal{X}}$ are the feasible sets. The objective function for combinatorial optimization problems is often linear, and referred to as **modular**. That is, we usually want to pick the feasible set with the largest possible weight. Later, we will study combinatorial analogs of convexity and concavity called submodularity and supermodularity (though the association is not that clear-cut). **Matroids** are the combinatorial analog of convex sets, and are useful in a variety of contexts.

1.1 Maximum Weight Forest

Recall the **maximum weight forest** problem: Given a graph $G = (V, E)$ and weights $w_e \in \mathbb{R}$ on each edge $e \in E$, find the maximum weight acyclic subgraph (forest) of G. This is a generalization of the MST problem. Let $|V| = n$ and $|E| = m$.

We can solve this problem with a greedy algorithm similar in implementation to Kruskal's Algorithm:

Algorithm 1 Greedy Algorithm for Maximum Weight Forest

```
Let $B = \emptyset$
Sort the non-negative weights in descending order of weight, i.e., $e_1, \dots, e_m$ with $w_1 \geq w_2 \geq \cdots \geq w_m \geq 0$
for $i = 1 \to m$ do
  if $B \cup (e_i)$ is acyclic then
    Set $B = B \cup (e_i)$
  end if
end for
```

Theorem 1. *The greedy algorithm outputs a maximum weight forest.*

Proof. We begin by establishing 3 critical properties of the greedy algorithm:

Lemma 2. *The greedy algorithm has the following properties:*

1. *The empty set is acyclic*
2. *If $A \subseteq E$ is acyclic, and $B \subseteq A$, then B is acyclic*
3. *If $A, B \subseteq E$ are acyclic and $|B| > |A|$, then $\exists e \in B \setminus A$ such that $A \cup \{e\}$ is acyclic*

Proof. It is easy to see that properties 1 and 2 are true, but property 3 is more subtle. Recall that if $C \subseteq E$ is acyclic, then $|C| = n - \mathit{components}(C)$ by induction. When $|B| > |A|$ and both are acyclic, this implies that $\mathit{components}(B) < \mathit{components}(A)$. It can't be that all $e \in B$ are contained within the connected components of A; there must be some $e \in B$ that connects two components of A, so we can choose this edge. Note that this also implies that there is at least one way to add edges from B to A such that $|B| = |A|$. □

We prove the theorem by induction. Let the inductive hypothesis for step i be that there exists a maximum weight forest B_i^* which "agrees" with the algorithm's choices on $e_1, \ldots, e_i$. That is, if B_i is the algorithm's choices up to i, then $B_i = B_i^* \cap \{e_1, \ldots, e_i\}$.

Assume the inductive hypothesis is true for step $i-1$ and consider step i. If $e \notin B_i$, then $B_{i-1} \cup \{e_i\}$ is cyclic. Since $B_{i-1} \subseteq B_{i-1}^*$, then $e_i \notin B_{i-1}^*$ by property 2. So, take $B_i^* = B_{i-1}^*$.

If $e_i \in B_i$ and $e_i \notin B_{i-1}^*$, then we build B_i^* by repeatedly extending B_i using B_{i-1}^* by property 3. Recall that B_i agrees with B_{i-1}^* on the first $i-1$ edges. So, $B_i^* = B_{i-1}^* \cup \{e_i\} \setminus \{e_k\}$ for some $k > i$. Since the edges are sorted in descending order of weight, we exchanged a lower weight edge for an edge of equal or greater weight, so B_i^* must be optimal. Therefore the greedy algorithm achieves the optimal solution at termination. □

1.2 Types of Matroids

Note that to prove optimality of the greedy algorithm, all we needed were the 3 key properties of the set system $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

1. $\emptyset \in \mathcal{I}$
2. If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$
3. If $A, B \in \mathcal{I}$ and $|B| > |A|$, then $\exists e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$. Recall that this implies that at least one way to add elements from B to A such that $|B| = |A|$. This is the most important property of a matroid, and is called the **exchange property**.

These are called the **matroid axioms** and the set system $\mathcal{M}$ is called a matroid if the axioms hold in the set system. In the previous section, we analyzed the matroid where $\mathcal{X} = E$ and $\mathcal{I} = \{\text{forests}\}$, called the graphic matroid. Some other common matroids include:

- Linear matroid: $\mathcal{X} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\} \in \mathbb{R}^n$ and $S \in \mathcal{I}$ if all $\boldsymbol{v}_i \in S$ are linearly independent. The matroid axioms are:
    1. Empty set is linearly independent
    2. Every subset of a linearly independent set is linearly independent
    3. All spans of $\mathbb{R}^n$ have size n
- Uniform Matroid: $\mathcal{X} = \{1, \ldots, n\}$ and $S \in \mathcal{I}$ if $|S| \leq k$ for a given $k \geq 0$. The matroid axioms are:
    1. Empty set has size ≥ 0
    2. Removing an element from a set does not increase its size
    3. If $|S| \leq |T| \leq k$, then $\exists e \in T \notin S$
- Partition Matroid: $\mathcal{X} = X_1 \cup \cdots \cup X_m$ and $S \in \mathcal{I}$ if S contains $\leq k_j$ elements from each X_j. This is just a generalization of the Uniform Matroid.
- Transversal Matroid: Given a bipartite graph $E \subseteq L \times R$, we have $\mathcal{X} = L$ and $S \in \mathcal{I}$ if there exists a bipartite matching on the nodes in S. Note that we don't actually encode the edges in the matroid.
    1. Empty set is a matching
    2. Removing a node preserves the matchings on the other nodes
    3. Like in the Ford-Fulkerson Algorithm, there is an augmenting path which can extend the matching of S to some $x \in T \setminus S$. This allows us to iteratively build a maximum matching, which is an easy proof of the Ford-Fulkerson Algorithm on bipartite graphs.

1.3 Terminology and Properties

Theorem 3. *The greedy algorithm returns the maximum weight feasible set for every choice of weights if and only if the set system $\mathcal{M} = (\mathcal{X}, \mathcal{I})$ is a matroid. Thus, matroids fully characterize the class of problems where the greedy algorithm is successful. Note that to implement the greedy algorithm on a matroid, we jut need an independence oracle to check whether the next highest weight element is in $\mathcal{I}$ or not. Then, the greedy algorithm runs in time $\mathcal{O}(n \log n) + nT$ where T is the runtime of the oracle.*

Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$. We introduce some terminology:

- A set $A \in \mathcal{I}$ is called an **independent set** of the matroid
- The maximal independent set in $\mathcal{M}$ is called a **base** of $\mathcal{M}$
- The maximal independent subset of $S \subseteq \mathcal{X}$ is called the **base** of S in $\mathcal{M}$; that is, it is the base of $\mathcal{M}$ after deleting $\bar{S}$
- The minimal dependent set of $\mathcal{X}$ is called a **circuit**; that is, a set where deleting one element creates an independent set

Lemma 4. *For every $S \subseteq \mathcal{X}$, all bases of S in $\mathcal{M}$ have equal cardinality, called it's **rank**. This follows from the exchange property. One example is that all bases of a subspace have rank equal to the dimension of the subspace.*

We can define the **rank function** of S with respect to $\mathcal{M}$ as $rank_{\mathcal{M}}(S) : 2^{\mathcal{X}} \to \mathbb{N}$. Given $S \subseteq \mathcal{X}$, $span(S) = \{i \in \mathcal{X} : rank(S) = rank(S \cup \{i\})\}$. That is, S itself plus elements which would form a circuit if added to the base of S. Note that $i \in \{1, \dots, n\}$ is selected by the greedy algorithm if and only if $i \notin span(\{1, \dots, i-1\})$.

1.4 Operations Preserving Matroidness

It is interesting to study operations preserving matroidness because they provide insight into complex problems. Below we detail some simple operations which preserve matroidness:

1. Deletion ($\mathcal{M} = \mathcal{M} \setminus B$)
2. Restriction ($\mathcal{M} = \mathcal{M} \setminus \bar{B}$)
3. Disjoint union of n matroids with disjoint $\mathcal{X}_i$
4. Contraction ($\mathcal{M} = \mathcal{M}/B$); here we can imagine B always included and consider independence relative to B, such as when we contract nodes into a supernode in a graphic matroid

There are others, including truncation, dual, and union, which we will cover later.