Announcements

- We should have heard from you about projects
- First two problems of HW3 released
  - It’s shorter, but still pace yourself.
Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets:

- Shortest paths
- Max-weight matching
- TSP
- ...

Related, directly or indirectly, to a large fraction of optimization problems in $P$ and also pops up in submodular/supermodular optimization problems.
Optimization over Sets

- Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets
  - Shortest paths
  - Max-weight matching
  - TSP
  - ...

- Set system: Pair $\mathcal{X}, \mathcal{I}$ where $\mathcal{X}$ is a finite ground set and $\mathcal{I} \subseteq 2^\mathcal{X}$ are the feasible sets

- Objective: often "linear", referred to as modular

- Analogues of concave convex: submodular and supermodular (in no particular order!)

- Today, we will look only at optimizing modular objectives over an extremely prolific family of set systems

- Related, directly or indirectly, to a large fraction of optimization problems in $\mathcal{P}$

- Also pops up in submodular/supermodular optimization problems
Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets:

- Shortest paths
- Max-weight matching
- TSP
- ...

Set system: Pair \((\mathcal{X}, \mathcal{I})\) where \(\mathcal{X}\) is a finite ground set and \(\mathcal{I} \subseteq 2^{\mathcal{X}}\) are the feasible sets.

Objective: often “linear”, referred to as modular.
Optimization over Sets

- Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets
  - Shortest paths
  - Max-weight matching
  - TSP
  - ...

- Set system: Pair \((X, I)\) where \(X\) is a finite ground set and \(I \subseteq 2^X\) are the feasible sets

- Objective: often “linear”, referred to as modular

- Analogues of concave convex: submodular and supermodular (in no particular order!)
Most combinatorial optimization problems can be thought of as choosing the best set from a family of allowable sets

- Shortest paths
- Max-weight matching
- TSP
- ...

Set system: Pair $(\mathcal{X}, \mathcal{I})$ where $\mathcal{X}$ is a finite *ground set* and $\mathcal{I} \subseteq 2^\mathcal{X}$ are the feasible sets

Objective: often “linear”, referred to as *modular*

Analogues of concave convex: *submodular* and *supermodular* (in no particular order!)

Today, we will look only at optimizing modular objectives over an extremely prolific family of set systems

- Related, directly or indirectly, to a large fraction of optimization problems in $P$
- Also pops up in submodular/supermodular optimization problems
Outline

1. Matroids and The Greedy Algorithm
2. Basic Terminology and Properties
3. The Matroid Polytope
4. Matroid Intersection
Maximum Weight Forest Problem

Given a connected undirected graph $G = (V, E)$, and weights $w_e \in \mathbb{R}$ on edges $e$, find a maximum weight acyclic subgraph (aka forest) of $G$.

- Slight generalization of minimum weight spanning tree
- We use $n$ and $m$ to denote $|V|$ and $|E|$, respectively.

Matroids and The Greedy Algorithm
The Greedy Algorithm

1. $B \leftarrow \emptyset$
2. Sort non-negative weight edges in decreasing order of weight $e_1, \ldots, e_m$, with $w_1 \geq w_2 \geq \ldots \geq w_m \geq 0$
3. For $i = 1$ to $m$:
   - if $B \cup \{e_i\}$ is acyclic, add $e_i$ to $B$. 

Theorem

The greedy algorithm outputs a maximum-weight forest.
The Greedy Algorithm

1. \( B \leftarrow \emptyset \)
2. Sort non-negative weight edges in decreasing order of weight
   - \( e_1, \ldots, e_m \), with \( w_1 \geq w_2 \geq \ldots \geq w_m \geq 0 \)
3. For \( i = 1 \) to \( m \):
   - if \( B \cup \{e_i\} \) is acyclic, add \( e_i \) to \( B \).

Theorem

The greedy algorithm outputs a maximum-weight forest.
Lemma

1. The empty set is acyclic
2. If $A$ is an acyclic set of edges, and $B \subseteq A$, then $B$ is also acyclic.
3. If $A, B$ are acyclic, and $|B| > |A|$, then there is $e \in B \setminus A$ such that $A \cup \{e\}$ is acyclic.
### Lemma

1. The empty set is acyclic.
2. If \( A \) is an acyclic set of edges, and \( B \subseteq A \), then \( B \) is also acyclic.
3. If \( A, B \) are acyclic, and \( |B| > |A| \), then there is \( e \in B \setminus A \) such that \( A \cup \{e\} \) is acyclic.

(1) and (2) are trivial, so let's prove (3)
Lemma

1. The empty set is acyclic
2. If $A$ is an acyclic set of edges, and $B \subseteq A$, then $B$ is also acyclic.

3. If $A, B$ are acyclic, and $|B| > |A|$, then there is $e \in B \setminus A$ such that $A \cup \{e\}$ is acyclic

Sub-lemma: if $C$ is acyclic, then $|C| = n - \#\text{components}(C)$.

- Induction
Lemma

1. The empty set is acyclic.
2. If $A$ is an acyclic set of edges, and $B \subseteq A$, then $B$ is also acyclic.
3. If $A, B$ are acyclic, and $|B| > |A|$, then there is $e \in B \setminus A$ such that $A \cup \{e\}$ is acyclic.

Sub-lemma: if $C$ is acyclic, then $|C| = n - \#\text{components}(C)$.

Induction

When $|B| > |A|$, this means $\#\text{components}(B) < \#\text{components}(A)$.
Lemma

1. The empty set is acyclic
2. If \( A \) is an acyclic set of edges, and \( B \subseteq A \), then \( B \) is also acyclic.
3. If \( A, B \) are acyclic, and \(|B| > |A|\), then there is \( e \in B \setminus A \) such that \( A \cup \{e\} \) is acyclic

Sub-lemma: if \( C \) is acyclic, then \(|C| = n - \#\text{components}(C)\).

- Induction
- When \(|B| > |A|\), this means \( \#\text{components}(B) < \#\text{components}(A) \)
- Can’t be that all \( e \in B \) are “inside” connected components of \( A \)
Lemma

1. The empty set is acyclic.
2. If \( A \) is an acyclic set of edges, and \( B \subseteq A \), then \( B \) is also acyclic.
3. If \( A, B \) are acyclic, and \( |B| > |A| \), then there is \( e \in B \setminus A \) such that \( A \cup \{e\} \) is acyclic.

Sub-lemma: if \( C \) is acyclic, then \( |C| = n - \#\text{components}(C) \).

- Induction
- When \( |B| > |A| \), this means \( \#\text{components}(B) < \#\text{components}(A) \)
- Can’t be that all \( e \in B \) are “inside” connected components of \( A \)
- Some \( e \in B \) must “go between” connected components of \( A \).
Lemma

1. The empty set is acyclic.
2. If \( A \) is an acyclic set of edges, and \( B \subseteq A \), then \( B \) is also acyclic.
   - Converse: if \( B \) cyclic then so is \( A \).
3. If \( A, B \) are acyclic, and \( |B| > |A| \), then there is \( e \in B \setminus A \) such that \( A \cup \{e\} \) is acyclic.
   - Inductively: can extend \( A \) by adding \( |B| - |A| \) elements from \( B \setminus A \).

Sub-lemma: if \( C \) is acyclic, then \( |C| = n - \text{#components}(C) \).
- Induction
- When \( |B| > |A| \), this means \( \text{#components}(B) < \text{#components}(A) \).
- Can’t be that all \( e \in B \) are “inside” connected components of \( A \).
- Some \( e \in B \) must “go between” connected components of \( A \).
Proof

Going back to proving the algorithm correct.

**Inductive Hypothesis (i)**

There is a maximum-weight acyclic forest $B_i^*$ which “agrees” with the algorithm’s choices on edges $e_1, \ldots, e_i$.

- i.e. if $B_i$ denotes the algorithm’s choice up to iteration $i$, then

  $$B_i = B_i^* \cap \{e_1, \ldots, e_i\}$$
Proof

Going back to proving the algorithm correct.

Inductive Hypothesis (i)

There is a maximum-weight acyclic forest $B_i^*$ which “agrees” with the algorithm’s choices on edges $e_1, \ldots, e_i$.

- i.e. if $B_i$ denotes the algorithm’s choice up to iteration $i$, then $B_i = B_i^* \cap \{e_1, \ldots, e_i\}$

- Assume true for step $i - 1$, and consider step $i$
Proof

Going back to proving the algorithm correct.

Inductive Hypothesis (i)

There is a maximum-weight acyclic forest $B_i^*$ which “agrees” with the algorithm’s choices on edges $e_1, \ldots, e_i$.

- i.e. if $B_i$ denotes the algorithm’s choice up to iteration $i$, then $B_i = B_i^* \cap \{e_1, \ldots, e_i\}$

- Assume true for step $i - 1$, and consider step $i$
- If $e_i \notin B_i$, then $B_{i-1} \cup \{e_i\}$ is cyclic. Since $B_{i-1} \subseteq B_{i-1}^*$, then $e_i \notin B_{i-1}^*$ (Property 2). So take $B_i^* = B_{i-1}^*$. 

Matroids and The Greedy Algorithm
Going back to proving the algorithm correct.

**Inductive Hypothesis (i)**

There is a maximum-weight acyclic forest $B_i^*$ which “agrees” with the algorithm’s choices on edges $e_1, \ldots, e_i$.

- i.e. if $B_i$ denotes the algorithm’s choice up to iteration $i$, then $B_i = B_i^* \cap \{e_1, \ldots, e_i\}$

Assume true for step $i - 1$, and consider step $i$

- If $e_i \notin B_i$, then $B_{i-1} \cup \{e_i\}$ is cyclic. Since $B_{i-1} \subseteq B_{i-1}^*$, then $e_i \notin B_{i-1}^*$ (Property 2). So take $B_i^* = B_{i-1}^*$.
- If $e_i \in B_i$ and $e_i \notin B_i^*$, extend $B_i$ to the size of $B_{i-1}^*$ (property 3)
  - Recall that $B_{i-1} = B_i \setminus \{e_i\} \subseteq B_{i-1}^*$
  - $B_i^* = B_{i-1}^* \cup \{e_i\} \setminus \{e_k\}$ for some $k > i$
  - $B_i^*$ has weight no less than $B_{i-1}^*$, so optimal.
To prove optimality of the greedy algorithm, all we needed was the following.

**Matroids**

A set system \( M = (\mathcal{X}, \mathcal{I}) \) is a **matroid** if

1. \( \emptyset \in \mathcal{I} \)
2. If \( A \in \mathcal{I} \) and \( B \subseteq A \), then \( B \in \mathcal{I} \) (Downward Closure)
3. If \( A, B \in \mathcal{I} \) and \( |B| > |A| \), then \( \exists x \in B \setminus A \) such that \( A \cup \{x\} \in \mathcal{I} \) (Exchange Property)
To prove optimality of the greedy algorithm, all we needed was the following.

**Matroids**

A set system $M = (X, \mathcal{I})$ is a matroid if

1. $\emptyset \in \mathcal{I}$
2. If $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$ (Downward Closure)
3. If $A, B \in \mathcal{I}$ and $|B| > |A|$, then $\exists x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$ (Exchange Property)

- Three conditions above are called the *matroid axioms*
- $A \in \mathcal{I}$ is called an *independent set* of the matroid.
To prove optimality of the greedy algorithm, all we needed was the following.

**Matroids**

A set system $M = (X, I)$ is a matroid if

1. $\emptyset \in I$
2. If $A \in I$ and $B \subseteq A$, then $B \in I$ (Downward Closure)
3. If $A, B \in I$ and $|B| > |A|$, then $\exists x \in B \setminus A$ such that $A \cup \{x\} \in I$ (Exchange Property)

- Three conditions above are called the matroid axioms
- $A \in I$ is called an independent set of the matroid.
- The matroid whose independent sets are acyclic subgraphs is called a graphic matroid
- Other examples abound!
Example: Linear Matroid

- $\mathcal{X}$ is a finite set of vectors $\{v_1, \ldots, v_m\} \subseteq \mathbb{R}^n$
- $S \in \mathcal{I}$ iff the vectors in $S$ are linearly independent

- Downward closure: If a set of vectors is linearly independent, then every subset of it is also
- Exchange property: Can always extend a low-dimension independent set $S$ by adding vectors from a higher dimension independent set $T$
Example: Uniform Matroid

- $\mathcal{X}$ is an arbitrary finite set $\{1, \ldots, n\}$.
- $S \in \mathcal{I}$ iff $|S| \leq k$.

- Downward closure: If a set $S$ has $|S| \leq k$ then the same holds for $T \subseteq S$.

- Exchange property: If $|S| < |T| \leq k$, then there is an element in $T \setminus S$, and we can add it to $S$ while preserving independence.
Example: Partition Matroid

- $\mathcal{X}$ is the disjoint union of classes $X_1, \ldots, X_m$
- Each class $X_j$ has an upperbound $k_j$.
- $S \in \mathcal{I}$ iff $|S \cap X_j| \leq k_j$ for all $j$

- This is the “disjoint union” of a number of uniform matroids
Example: Transversal Matroid

- Described by a bipartite graph $E \subseteq L \times R$
- $\mathcal{X} = L$
- $S \in \mathcal{I}$ iff there is a bipartite matching which matches $S$

- Downward closure: If we can match $S$, then we can match $T \subseteq S$.
- Exchange property: If $|T| > |S|$ is matchable, then an augmenting path/alternating path amends the extends the matching of $S$ to some $x \in T \setminus S$. 
The Greedy Algorithm on Matroids

**The Greedy Algorithm**

1. $B \leftarrow \emptyset$
2. Sort nonnegative elements of $\mathcal{X}$ in decreasing order of weight
   - $\{1, \ldots, n\}$ with $w_1 \geq w_2, \geq \ldots \geq w_n \geq 0$.
3. For $i = 1$ to $n$:
   - if $B \cup \{i\} \in \mathcal{I}$, add $i$ to $B$.

**Theorem**

_The greedy algorithm returns the maximum weight set for every choice of weights if and only if the set system $(\mathcal{X}, \mathcal{I})$ is a matroid._

We already saw the “if” direction. We will skip “only if”.
The Greedy Algorithm on Matroids

The Greedy Algorithm

1. \( B \leftarrow \emptyset \)
2. Sort nonnegative elements of \( X \) in decreasing order of weight
   - \( \{1, \ldots, n\} \) with \( w_1 \geq w_2 \geq \ldots \geq w_n \geq 0 \).
3. For \( i = 1 \) to \( n \):
   - if \( B \cup \{i\} \in \mathcal{I} \), add \( i \) to \( B \).
The Greedy Algorithm

1. $B \leftarrow \emptyset$
2. Sort nonnegative elements of $\mathcal{X}$ in decreasing order of weight
   - $\{1, \ldots, n\}$ with $w_1 \geq w_2 \geq \ldots \geq w_n \geq 0$.
3. For $i = 1$ to $n$:
   - if $B \cup \{i\} \in \mathcal{I}$, add $i$ to $B$.

To implement this, we need an independence oracle for step 3
- A subroutine which checks whether $S \in \mathcal{I}$ or not.
- Runs in time $O(n \log n) + nT$, where $T$ is runtime of the independence oracle.
The Greedy Algorithm on Matroids

The Greedy Algorithm

1. $B \leftarrow \emptyset$
2. Sort nonnegative elements of $X$ in decreasing order of weight
   - $\{1, \ldots, n\}$ with $w_1 \geq w_2 \geq \ldots \geq w_n \geq 0$.
3. For $i = 1$ to $n$:
   - if $B \cup \{i\} \in \mathcal{I}$, add $i$ to $B$.

To implement this, we need an independence oracle for step 3
- A subroutine which checks whether $S \in \mathcal{I}$ or not.
- Runs in time $O(n \log n) + nT$, where $T$ is runtime of the
  independence oracle.
- For most “natural” matroids, independence oracle is easy to
  implement efficiently
  - Graphic matroid
  - Linear matroid
  - Uniform/partition matroid
  - Transversal matroid
1. Matroids and The Greedy Algorithm
2. Basic Terminology and Properties
3. The Matroid Polytope
4. Matroid Intersection
Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

- An **independent set** is a set $A \in \mathcal{I}$. 

---

**Basic Terminology and Properties 13/30**
Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

- An **independent set** is a set $A \in \mathcal{I}$.
- A **base** of $\mathcal{M}$ is a maximal independent set.

**What are these for:**
- Graphic matroid
- Linear matroid
- Uniform matroid
- Partition matroid
- Transversal matroid
Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

- An **independent set** is a set $A \in \mathcal{I}$.
- A **base** of $\mathcal{M}$ is a maximal independent set.
- A **base of** $S \subseteq \mathcal{X}$ in $\mathcal{M}$ is maximal independent subset of $S$.
  - I.e. a base of the matroid after deleting $\overline{S}$.

**Graphic matroid**

**Linear matroid**

**Uniform matroid**

**Partition matroid**

**Transversal matroid**
Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

- An **independent set** is a set $A \in \mathcal{I}$.
- A **base** of $\mathcal{M}$ is a maximal independent set.
- A **base of** $S \subseteq \mathcal{X}$ in $\mathcal{M}$ is a maximal independent subset of $S$.
  - I.e. a base of the matroid after deleting $\overline{S}$.
- A **circuit** $S \subseteq \mathcal{X}$ is a minimal dependent subset of $\mathcal{X}$.
Consider a matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$.

- An **independent set** is a set $A \in \mathcal{I}$.
- A **base** of $\mathcal{M}$ is a maximal independent set.
- A **base of $S \subseteq \mathcal{X}$ in $\mathcal{M}$** is maximal independent subset of $S$.
  - I.e. a base of the matroid after deleting $\overline{S}$.
- A **circuit $S \subseteq \mathcal{X}$** is a minimal dependent subset of $\mathcal{X}$.

What are these for:
- Graphic matroid
- Linear matroid
- Uniform matroid
- Partition matroid
- Transversal matroid
Rank

Lemma

For every $S \subseteq \mathcal{X}$, all bases of $S$ in $\mathcal{M}$ have the same cardinality.

- Special case of $S = \mathcal{X}$: all bases of $\mathcal{M}$ have the same cardinality.
- Should remind you of vector space dimension
For every $S \subseteq \mathcal{X}$, all bases of $S$ in $\mathcal{M}$ have the same cardinality.

- Special case of $S = \mathcal{X}$: all bases of $\mathcal{M}$ have the same cardinality.
- Should remind you of vector space dimension

Follows directly from the exchange property.
Rank

Lemma

For every $S \subseteq \mathcal{X}$, all bases of $S$ in $\mathcal{M}$ have the same cardinality.

- Special case of $S = \mathcal{X}$: all bases of $\mathcal{M}$ have the same cardinality.
- Should remind you of vector space dimension

- Follows directly from the exchange property.

The following analogue of vector space dimension is well-defined.

Rank

- The Rank of $S \subseteq \mathcal{X}$ in $\mathcal{M}$ is the size of the maximal independent subsets (i.e. bases) of $S$.
- The rank of $\mathcal{M}$ is the size of the bases of $\mathcal{M}$.
- The function $\text{rank}_\mathcal{M}(S) : 2^\mathcal{X} \rightarrow \mathbb{N}$ is called the rank function of $\mathcal{M}$. 

Basic Terminology and Properties 14/30
## Lemma

For every $S \subseteq \mathcal{X}$, all bases of $S$ in $\mathcal{M}$ have the same cardinality.

- Special case of $S = \mathcal{X}$: all bases of $\mathcal{M}$ have the same cardinality.
- Should remind you of vector space dimension

- Follows directly from the exchange property.

The following analogue of vector space dimension is well-defined.

## Rank

- The **Rank** of $S \subseteq \mathcal{X}$ in $\mathcal{M}$ is the size of the maximal independent subsets (i.e. bases) of $S$.
- The rank of $\mathcal{M}$ is the size of the bases of $\mathcal{M}$.
- The function $\text{rank}_\mathcal{M}(S) : 2^{\mathcal{X}} \rightarrow \mathbb{N}$ is called the rank function of $\mathcal{M}$.

E.g.: Graphic matroid, linear matroid, transversal matroid
Span

Given $S \subseteq \mathcal{X}$, $span(S) = \{ i \in \mathcal{X} : rank(S) = rank(S \cup \{i\}) \}$

- i.e. the elements which would form a circuit if added to a base of $S$
- e.g.: Linear matroid, graphic matroid, uniform matroid.
Given \( S \subseteq X \), \( \text{span}(S) = \{ i \in X : \text{rank}(S) = \text{rank}(S \cup \{i\}) \} \)

- i.e. the elements which would form a circuit if added to a base of \( S \)
- e.g.: Linear matroid, graphic matroid, uniform matroid.

Observation

\( i \) is selected by the greedy algorithm iff \( i \not\in \text{span}(\{1, \ldots, i-1\}) \)
Operations preserving Matroidness

Given $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, consider the following operations:

- **Deletion**: For $B \subseteq \mathcal{X}$, we define $\mathcal{M} \setminus B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = X \setminus B$,
  \[
  \mathcal{I}' = \{ S \subseteq X' : S \in \mathcal{I} \}
  \]

- **Graphic**: deleting edges from the graph

- **Disjoint union**: Given $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{I}_2)$ and $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{I}_2)$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, we define $\mathcal{M}_1 \oplus \mathcal{M}_2 = (\mathcal{X}_1 \cup \mathcal{X}_2, \{ A_1 \cup A_2 : A_1 \in \mathcal{I}_1, A_2 \in \mathcal{I}_2 \})$

  **Graphic**: combining two node-disjoint graphs

- **Contraction**: Given $B \subseteq \mathcal{X}$, let $\mathcal{M}/B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = X \setminus B$,
  \[
  \mathcal{I}' = \{ S \subseteq X' : B \cup S \in \mathcal{I} \}
  \]

  i.e. Think of $B$ as always being included

  **Graphic**: contract the connected components of $B$

- Others: truncation, dual, union...
Operations preserving Matroidness

Given $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, consider the following operations:

- **Deletion**: For $B \subseteq \mathcal{X}$, we define $\mathcal{M} \setminus B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = \mathcal{X} \setminus B$, 
  \[ \mathcal{I}' = \{ S \subseteq \mathcal{X}' : S \in \mathcal{I} \} \]

- **Graphic**: deleting edges from the graph

- **Disjoint union**: Given $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{I}_2)$ and $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{I}_2)$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, we define
  \[ \mathcal{M}_1 \oplus \mathcal{M}_2 = (\mathcal{X}_1 \cup \mathcal{X}_2, \{ A_1 \cup A_2 : A_1 \in \mathcal{I}_1, A_2 \in \mathcal{I}_2 \}) \]

- **Graphic**: combining two node-disjoint graphs

Others: truncation, dual, union...
Operations preserving Matroidness

Given $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, consider the following operations:

- **Deletion**: For $B \subseteq \mathcal{X}$, we define $\mathcal{M} \setminus B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = \mathcal{X} \setminus B$,
  \[ \mathcal{I}' = \{ S \subseteq \mathcal{X}' : S \in \mathcal{I} \} \]
  - Graphic: deleting edges from the graph

- **Disjoint union**: Given $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{I}_2)$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, we define
  \[ \mathcal{M}_1 \oplus \mathcal{M}_2 = (\mathcal{X}_1 \cup \mathcal{X}_2, \{ A_1 \cup A_2 : A_1 \in \mathcal{I}_1, A_2 \in \mathcal{I}_2 \}) \]
  - Graphic: combining two node-disjoint graphs

- **Contraction**: Given $B \subseteq \mathcal{X}$, let $\mathcal{M}/B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = \mathcal{X} \setminus B$,
  \[ \mathcal{I}' = \{ S \subseteq \mathcal{X}' : B \cup S \in \mathcal{I} \} \]
  - i.e. Think of $B$ as always being included
  - Graphic: contract the connected components of $B

Others: truncation, dual, union...
Operations preserving Matroidness

Given $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, consider the following operations:

- **Deletion**: For $B \subseteq \mathcal{X}$, we define $\mathcal{M} \setminus B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = \mathcal{X} \setminus B$,
  \[ \mathcal{I}' = \{ S \subseteq \mathcal{X}' : S \in \mathcal{I} \} \]
  - Graphic: deleting edges from the graph

- **Disjoint union**: Given $\mathcal{M}_ = (\mathcal{X}_1, \mathcal{I}_2)$ and $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{I}_2)$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, we define
  \[ \mathcal{M}_1 \oplus \mathcal{M}_2 = (\mathcal{X}_1 \cup \mathcal{X}_2, \{ A_1 \cup A_2 : A_1 \in \mathcal{I}_1, A_2 \in \mathcal{I}_2 \}) \]
  - Graphic: combining two node-disjoint graphs

- **Contraction**: Given $B \subseteq \mathcal{X}$, let $\mathcal{M}/B = (\mathcal{X}', \mathcal{I}')$ with $\mathcal{X}' = \mathcal{X} \setminus B$,
  \[ \mathcal{I}' = \{ S \subseteq \mathcal{X}' : B \cup S \in \mathcal{I} \} \]
  - i.e. Think of $B$ as always being included
  - Graphic: contract the connected components of $B$

- **Others**: truncation, dual, union...
Optimization over matroids is “easy”, in the same way that optimization over convex sets is “easy”
Optimization over matroids is “easy”, in the same way that optimization over convex sets is “easy”

Operations preserving set convexity are analogous to operations preserving matroid structure
Optimization over matroids is “easy”, in the same way that optimization over convex sets is “easy”.

Operations preserving set convexity are analogous to operations preserving matroid structure.

Arguably, matroids and submodular functions are discrete analogues of convex sets and convex functions, respectively.

- Less exhaustive
1. Matroids and The Greedy Algorithm

2. Basic Terminology and Properties

3. The Matroid Polytope

4. Matroid Intersection
As is often the case with tractable discrete problems, we can view their feasible set as a polyhedron.
As is often the case with tractable discrete problems, we can view their feasible set as a polyhedron.

For $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, the convex hull of independent sets can be written as a polytope in a natural way:

- The polytope is “solvable”, and admits a polytime separation oracle.
As is often the case with tractable discrete problems, we can view their feasible set as a polyhedron.

For $\mathcal{M} = (\mathcal{X}, \mathcal{I})$, the convex hull of independent sets can be written as a polytope in a natural way:
- The polytope is “solvable”, and admits a polytime separation oracle.

This perspective will be crucial for more advanced applications of matroids:
- Optimization of linear functions over matroid intersections
- Optimization of submodular functions over matroids
The Matroid Polytope

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]
\[
x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

- Assigns a variable $x_i$ to every element $i$ of the ground set
- Each feasible $x$ is a fractional subset of $\mathcal{X}$
  - $0 \leq x_i \leq 1$ since the rank of a singleton is at most 1.
The Matroid Polytope

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]
\[
x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

- Assigns a variable $x_i$ to every element $i$ of the ground set
- Each feasible $x$ is a fractional subset of $\mathcal{X}$
  - $0 \leq x_i \leq 1$ since the rank of a singleton is at most 1.
- The 0-1 indicator vector $x_I$ for independent set $I \in \mathcal{I}$ is in the above polytope
Assigns a variable $x_i$ to every element $i$ of the ground set
Each feasible $x$ is a fractional subset of $\mathcal{X}$
  - $0 \leq x_i \leq 1$ since the rank of a singleton is at most 1.
The 0-1 indicator vector $x_I$ for independent set $I \in \mathcal{I}$ is in the above polytope
In fact, we will show that $\mathcal{P}(\mathcal{M})$ is precisely the convex hull of independent sets $\mathcal{I}$
**The Matroid Polytope**

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]
\[
x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

- Assigns a variable $x_i$ to every element $i$ of the ground set
- Each feasible $x$ is a fractional subset of $\mathcal{X}$
  - $0 \leq x_i \leq 1$ since the rank of a singleton is at most 1.
- The 0-1 indicator vector $x_I$ for independent set $I \in \mathcal{I}$ is in the above polytope
- In fact, we will show that $\mathcal{P}(\mathcal{M})$ is precisely the convex hull of independent sets $\mathcal{I}$
- Note: polytope has $2^{|\mathcal{X}|}$ constraints.
Integrality of the Matroid Polytope

Polytope $P(M)$ for $M = (X, I)$

$$\sum_{i \in S} x_i \leq rank_M(S), \quad \text{for } S \subseteq X.$$  
$$x_i \geq 0, \quad \text{for } i \in X.$$  

Theorem

$$P(M) = \text{convexhull} \{x_I : I \in I\}$$
Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

$$\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.$$  

$$x_i \geq 0, \quad \text{for } i \in \mathcal{X}.$$  

Theorem

$$\mathcal{P}(\mathcal{M}) = \text{convexhull} \{x_I : I \in \mathcal{I}\}$$

- It is clear that $\mathcal{P}(\mathcal{M}) \supseteq \text{convexhull} \{x_I : I \in \mathcal{I}\}$
Integrality of the Matroid Polytope

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

$$\sum_{i \in S} x_i \leq \text{rank}_{\mathcal{M}}(S), \quad \text{for } S \subseteq \mathcal{X}.$$  

$$x_i \geq 0, \quad \text{for } i \in \mathcal{X}.$$  

Theorem

$\mathcal{P}(\mathcal{M}) = \text{convexhull} \{x_I : I \in \mathcal{I}\}$

- It is clear that $\mathcal{P}(\mathcal{M}) \supseteq \text{convexhull} \{x_I : I \in \mathcal{I}\}$
- To show that $\mathcal{P}(\mathcal{M}) \subseteq \text{convexhull} \{x_I : I \in \mathcal{I}\}$, we will show that every vertex of $\mathcal{P}(\mathcal{M})$ equals $x_I$ for some $I \in \mathcal{I}$.
Integrality of the Matroid Polytope

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

$$\sum_{i \in S} x_i \leq rank_{\mathcal{M}}(S), \quad \text{for } S \subseteq \mathcal{X}.$$  
$$x_i \geq 0, \quad \text{for } i \in \mathcal{X}.$$  

Theorem

$$\mathcal{P}(\mathcal{M}) = \text{convexhull}\{x_I : I \in \mathcal{I}\}$$

- It is clear that $\mathcal{P}(\mathcal{M}) \supseteq \text{convexhull}\{x_I : I \in \mathcal{I}\}$
- To show that $\mathcal{P}(\mathcal{M}) \subseteq \text{convexhull}\{x_I : I \in \mathcal{I}\}$, we will show that every vertex of $\mathcal{P}(\mathcal{M})$ equals $x_I$ for some $I \in \mathcal{I}$.
- Recall: suffices to show that every linear function $w^T x$ is maximized over $\mathcal{P}(\mathcal{M})$ at some $x_I$ for $I \in \mathcal{I}$. 

The Matroid Polytope
Recall: The Greedy Algorithm

1. \( B \leftarrow \emptyset \)
2. Sort nonnegative elements of \( \mathcal{X} \) in decreasing order of weight with \( \{1, \ldots, n \} \) with \( w_1 \geq w_2, \geq \ldots \geq w_n \geq 0 \).
3. For \( i = 1 \) to \( n \):
   - if \( B \cup \{i\} \in \mathcal{I} \), add \( i \) to \( B \).

Theorem

The greedy algorithm returns the maximum weight set for every choice of weights if and only if the set system \((\mathcal{X}, \mathcal{I})\) is a matroid.
Recall: The Greedy Algorithm

1. \( B \leftarrow \emptyset \)
2. Sort nonnegative elements of \( X \) in decreasing order of weight
   \( \{1, \ldots, n\} \with w_1 \geq w_2 \geq \ldots \geq w_n \geq 0 \).
3. For \( i = 1 \) to \( n \):
   - if \( B \cup \{i\} \in \mathcal{I} \), add \( i \) to \( B \).

Theorem

The greedy algorithm returns the maximum weight set for every choice of weights if and only if the set system \((X, \mathcal{I})\) is a matroid.

- We can think of the greedy algorithm as computing the indicator vector \( x^* = x_B \in \mathcal{P}(\mathcal{M}) \).
- We will show that \( x^* \) maximizes \( w^T x \) over \( x \in \mathcal{P}(\mathcal{M}) \).
Recall: Observation

$i$ is selected by the greedy algorithm iff $i \not\in \text{span}(\{1, \ldots, i - 1\})$

- i.e. if $\text{rank}[1 : i] - \text{rank}[1 : i - 1] = 1$. 

Consider an arbitrary $x \in \mathcal{P}(\mathcal{M})$

$$\sum_i w_i x_i = \sum_i (w_i - w_i + 1) x(1 : i) \leq \sum_i (w_i - w_i + 1) \text{rank}(1 : i) = \sum_i w_i (\text{rank}[1 : i] - \text{rank}[1 : i - 1])$$
Recall: Observation

\[ i \text{ is selected by the greedy algorithm iff } i \not\in \text{span}(\{1, \ldots, i - 1\}) \]

- i.e. if \( \text{rank}[1 : i] - \text{rank}[1 : i - 1] = 1 \).

Therefore, \( x_i^* = \text{rank}[1 : i] - \text{rank}[1 : i - 1] \)

\[
\sum_i w_i x_i^* = \sum_i w_i (\text{rank}[1 : i] - \text{rank}[1 : i - 1])
\]
Recall: Observation

$i$ is selected by the greedy algorithm iff $i \not\in \text{span}(\{1, \ldots, i-1\})$

- i.e. if $\text{rank}[1 : i] - \text{rank}[1 : i - 1] = 1$.

Therefore, $x_i^* = \text{rank}[1 : i] - \text{rank}[1 : i - 1]$

$$\sum_i w_i x_i^* = \sum_i w_i (\text{rank}[1 : i] - \text{rank}[1 : i - 1])$$

Consider an arbitrary $x \in \mathcal{P}(\mathcal{M})$

$$\sum_i w_i x_i = \sum_i w_i x_i^*$$
Recall: Observation

\( i \) is selected by the greedy algorithm iff \( i \not\in \text{span}(\{1, \ldots, i - 1\}) \)

- i.e. if \( \text{rank}[1 : i] - \text{rank}[1 : i - 1] = 1 \).

Therefore, \( x_i^* = \text{rank}[1 : i] - \text{rank}[1 : i - 1] \)

\[
\sum_i w_i x_i^* = \sum_i w_i (\text{rank}[1 : i] - \text{rank}[1 : i - 1])
\]

Consider an arbitrary \( x \in \mathcal{P}(\mathcal{M}) \)

\[
\sum_i w_i x_i = \sum_i (w_i - w_{i+1}) x(1 : i)
\]
Recall: Observation

\( i \) is selected by the greedy algorithm iff \( i \not\in \text{span}(\{1, \ldots, i - 1\}) \)

\( \text{i.e. if } \text{rank}[1 : i] - \text{rank}[1 : i - 1] = 1. \)

Therefore, \( x_i^* = \text{rank}[1 : i] - \text{rank}[1 : i - 1] \)

\[ \sum_i w_i x_i^* = \sum_i w_i (\text{rank}[1 : i] - \text{rank}[1 : i - 1]) \]

Consider an arbitrary \( x \in \mathcal{P}(\mathcal{M}) \)

\[ \sum_i w_i x_i = \sum_i (w_i - w_{i+1}) x(1 : i) \leq \sum_i (w_i - w_{i+1}) \text{rank}(1 : i) \]
Recall: Observation

\( i \) is selected by the greedy algorithm iff \( i \not\in \text{span}(\{1, \ldots, i-1\}) \)

- i.e. if \( \text{rank}[1:i] - \text{rank}[1:i-1] = 1 \).

Therefore, \( x_i^* = \text{rank}[1:i] - \text{rank}[1:i-1] \)

\[
\sum_i w_i x_i^* = \sum_i w_i (\text{rank}[1:i] - \text{rank}[1:i-1])
\]

Consider an arbitrary \( x \in \mathcal{P}(\mathcal{M}) \)

\[
\sum_i w_i x_i = \sum_i (w_i - w_{i+1}) x(1:i) \\
\leq \sum_i (w_i - w_{i+1}) \text{rank}(1:i) \\
= \sum_i w_i (\text{rank}[1:i] - \text{rank}[1:i-1])
\]
The Matroid Base Polytope

- The matroid polytope is the convex hull of independent sets
  - Graphic: convex hull of forests
- What if we wish to consider only “full-rank” sets?
  - Graphic: spanning trees
The Matroid Base Polytope

- The matroid polytope is the convex hull of independent sets
  - Graphic: convex hull of forests
- What if we wish to consider only “full-rank” sets?
  - Graphic: spanning trees

Polytope $P_{base}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\begin{align*}
\sum_{i \in S} x_i & \leq rank_{\mathcal{M}}(S), \quad \text{for } S \subseteq \mathcal{X}. \\
\sum_{i \in \mathcal{X}} x_i & = rank(\mathcal{M}) \\
x_i & \geq 0, \quad \text{for } i \in \mathcal{X}.
\end{align*}
\]

- The 0-1 indicator vector $x_B$ for a base $B$ of $\mathcal{M}$ is in above polytope
The Matroid Base Polytope

- The matroid polytope is the convex hull of independent sets
  - Graphic: convex hull of forests
- What if we wish to consider only “full-rank” sets?
  - Graphic: spanning trees

**Polytope $\mathcal{P}_{\text{base}}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$**

$$\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.$$  

$$\sum_{i \in \mathcal{X}} x_i = \text{rank}(\mathcal{M})$$

$$x_i \geq 0, \quad \text{for } i \in \mathcal{X}.$$  

- The 0-1 indicator vector $x_B$ for a base $B$ of $\mathcal{M}$ is in above polytope
- In fact, we will show that $\mathcal{P}(\mathcal{M})$ is precisely the convex hull of bases of $\mathcal{M}$
Polytope $P_{\text{base}}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\begin{align*}
\sum_{i \in S} x_i &\leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}. \\
\sum_{i \in \mathcal{X}} x_i &\leq \text{rank}(\mathcal{M}) \\
x_i &\geq 0, \quad \text{for } i \in \mathcal{X}.
\end{align*}

Theorem

$P_{\text{base}}(\mathcal{M}) = \text{convexhull}\ \{x_B : B \text{ is a base of } \mathcal{M}\}$
Polytope $\mathcal{P}_{\text{base}}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]
\[
\sum_{i \in \mathcal{X}} x_i = \text{rank}(\mathcal{M})
\]
\[
x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

Theorem

$\mathcal{P}_{\text{base}}(\mathcal{M}) = \text{convexhull} \{ x_B : B \text{ is a base of } \mathcal{M} \}$

As before, one direction is obvious:

$\mathcal{P}_{\text{base}}(\mathcal{M}) \supseteq \text{convexhull} \{ x_B : B \text{ is a base of } \mathcal{M} \}$
The Matroid Polytope

Polytope \( P_{\text{base}}(\mathcal{M}) \) for \( \mathcal{M} = (\mathcal{X}, \mathcal{I}) \)

\[
\begin{align*}
\sum_{i \in S} x_i & \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}. \\
\sum_{i \in \mathcal{X}} x_i & = \text{rank}(\mathcal{M}) \\
x_i & \geq 0, \quad \text{for } i \in \mathcal{X}.
\end{align*}
\]

Theorem

\[ P_{\text{base}}(\mathcal{M}) = \text{convexhull} \{ x_B : B \text{ is a base of } \mathcal{M} \} \]

- As before, one direction is obvious:
  \[ P_{\text{base}}(\mathcal{M}) \supseteq \text{convexhull} \{ x_B : B \text{ is a base of } \mathcal{M} \} \]
- For the other direction, take \( x \in P_{\text{base}}(\mathcal{M}) \)
Polytope $\mathcal{P}_{\text{base}}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]

\[
\sum_{i \in \mathcal{X}} x_i = \text{rank}(\mathcal{M})
\]

\[x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

Theorem

$\mathcal{P}_{\text{base}}(\mathcal{M}) = \text{convexhull}\ \{x_B : B \text{ is a base of } \mathcal{M}\}$

- As before, one direction is obvious:
  \[\mathcal{P}_{\text{base}}(\mathcal{M}) \supseteq \text{convexhull}\ \{x_B : B \text{ is a base of } \mathcal{M}\}\]
- For the other direction, take $x \in \mathcal{P}_{\text{base}}(\mathcal{M})$
- Since $x \in \mathcal{P}(\mathcal{M})$, $x$ is a convex combination of independent sets $I_1, \ldots, I_k$ of $\mathcal{M}$. 
Polytope \( P_{\text{base}}(\mathcal{M}) \) for \( \mathcal{M} = (\mathcal{X}, \mathcal{I}) \)

\[
\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.
\]

\[
\sum_{i \in \mathcal{X}} x_i = \text{rank}(\mathcal{M})
\]

\[
x_i \geq 0, \quad \text{for } i \in \mathcal{X}.
\]

**Theorem**

\( P_{\text{base}}(\mathcal{M}) = \text{convexhull } \{x_B : B \text{ is a base of } \mathcal{M}\} \)

- As before, one direction is obvious:
  \( P_{\text{base}}(\mathcal{M}) \supseteq \text{convexhull } \{x_B : B \text{ is a base of } \mathcal{M}\} \)

- For the other direction, take \( x \in P_{\text{base}}(\mathcal{M}) \)

- Since \( x \in \mathcal{P}(\mathcal{M}) \), \( x \) is a convex combination of independent sets \( I_1, \ldots, I_k \) of \( \mathcal{M} \).

- Since \( \|x\|_1 = \text{rank}(\mathcal{M}) \), and \( \|x_{I_\ell}\|_1 \leq \text{rank}(\mathcal{M}) \) for all \( \ell \), it must be that \( \|x_{I_1}\|_1 = \|x_{I_2}\|_1 = \ldots = \|x_{I_k}\|_1 = \text{rank}(\mathcal{M}) \)
When given an independence oracle for $\mathcal{M}$, we can maximize linear functions over $\mathcal{P}(\mathcal{M})$ in $O(n \log n)$ time.

- By integrality, same as finding max-weight independent set of $\mathcal{M}$. 

The Matroid Polytope
Solvability of Matroid Polytopes

Polytope $\mathcal{P}(\mathcal{M})$ for $\mathcal{M} = (\mathcal{X}, \mathcal{I})$

$$\sum_{i \in S} x_i \leq \text{rank}_\mathcal{M}(S), \quad \text{for } S \subseteq \mathcal{X}.$$  

$$x_i \geq 0, \quad \text{for } i \in \mathcal{X}.$$  

- When given an independence oracle for $\mathcal{M}$, we can maximize linear functions over $\mathcal{P}(\mathcal{M})$ in $O(n \log n)$ time.
  - By integrality, same as finding max-weight independent set of $\mathcal{M}$.

- Therefore, by equivalence of separation and optimization, can also implement a separation oracle for $\mathcal{P}(\mathcal{M})$.  

Solvability of Matroid Polytopes

Polytope $P(M)$ for $M = (X, I)$

\[ \sum_{i \in S} x_i \leq rank_M(S), \quad \text{for } S \subseteq X. \]

\[ x_i \geq 0, \quad \text{for } i \in X. \]

- When given an independence oracle for $M$, we can maximize linear functions over $P(M)$ in $O(n \log n)$ time
  - By integrality, same as finding max-weight independent set of $M$.
- Therefore, by equivalence of separation and optimization, can also implement a separation oracle for $P(M)$
- A more direct proof: reduces to submodular function minimization
  - $rank_M$ is a submodular set function.
Outline

1. Matroids and The Greedy Algorithm
2. Basic Terminology and Properties
3. The Matroid Polytope
4. Matroid Intersection
Matroid Intersection

- Optimization of linear functions over matroids is tractable
- Matroid operations provide an algebra for constructing new matroids from old
- We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

In general, does not produce a matroid
Nevertheless, it will turn out that maximizing linear functions over a matroid intersection is tractable
However, maximizing linear functions over the intersection of 3 or more matroids is NP-hard.
Optimization of linear functions over matroids is tractable.
Matroid operations provide an algebra for constructing new matroids from old.
We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

Matroid Intersection

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.
Matroid Intersection

- Optimization of linear functions over matroids is tractable
- Matroid operations provide an algebra for constructing new matroids from old
- We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

Matroid Intersection

Given matroids $\mathcal{M}_1 = (\mathcal{X}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_2)$ on the same ground set, we define the set system $\mathcal{M}_1 \cap \mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_1 \cap \mathcal{I}_2)$.

- i.e. a set is feasible if it is independent in both matroids
Optimization of linear functions over matroids is tractable
Matroid operations provide an algebra for constructing new matroids from old
We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

**Matroid Intersection**

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

- i.e. a set is feasible if it is independent in both matroids
- In general, does not produce a matroid
Matroid Intersection

- Optimization of linear functions over matroids is tractable
- Matroid operations provide an algebra for constructing new matroids from old
- We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

Matroid Intersection

Given matroids $\mathcal{M}_1 = (\mathcal{X}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_2)$ on the same ground set, we define the set system $\mathcal{M}_1 \cap \mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_1 \cap \mathcal{I}_2)$.

- i.e. a set is feasible if it is independent in both matroids
- In general, does not produce a matroid
- Nevertheless, it will turn out that maximizing linear functions over a matroid intersection is tractable
Optimization of linear functions over matroids is tractable

Matroid operations provide an algebra for constructing new matroids from old

We will look at one operation on matroids which does not produce a matroid, but nevertheless produces a solvable problem.

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

i.e. a set is feasible if it is independent in both matroids

In general, does not produce a matroid

Nevertheless, it will turn out that maximizing linear functions over a matroid intersection is tractable

However, maximizing linear functions over the intersection of 3 or more matroids is NP-hard
**Bipartite Matching**

Given a bipartite graph $G$, a set of edges $F$ is a bipartite matching if and only if each node is incident on at most one edge in $F$. 
Examples

Bipartite Matching
Given a bipartite graph $G$, a set of edges $F$ is a bipartite matching if and only if each node is incident on at most one edge in $F$.

Arborescence
Given a directed graph $G$, a set of edges is an $r$-arborescence is a tree directed away from the root $r$. 
Examples

Bipartite Matching
Given a bipartite graph $G$, a set of edges $F$ is a bipartite matching if and only if each node is incident on at most one edge in $F$.

Arborescence
Given a directed graph $G$, a set of edges is an $r$-arborescence is a tree directed away from the root $r$.

- Others: orientations of graphs, colorful spanning trees, ...
Matroid Intersection

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

- Optimizing a modular function over $M_1 \cap M_2$ is equivalent to optimizing a linear function over $\text{convexhull} \{x_I : I \in I_1 \cap I_2\}$.
- As it turns out, this is a solvable polytope.

Theorem

$$\mathcal{P}(M_1) \cap \mathcal{P}(M_2) = \text{convexhull} \{x_I : I \in I_1 \cap I_2\}$$
The Matroid Intersection Polytope

Matroid Intersection

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

- Optimizing a modular function over $M_1 \cap M_2$ is equivalent to optimizing a linear function over $\text{convexhull} \{x_I : I \in I_1 \cap I_2\}$.
- As it turns out, this is a solvable polytope.

Theorem

$\mathcal{P}(M_1) \cap \mathcal{P}(M_2) = \text{convexhull} \{x_I : I \in I_1 \cap I_2\}$

- One direction is obvious:
  $\mathcal{P}(M_1) \cap \mathcal{P}(M_2) \supseteq \text{convexhull} \{x_I : I \in I_1 \cap I_2\}$
Matroid Intersection

Given matroids $\mathcal{M}_1 = (\mathcal{X}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_2)$ on the same ground set, we define the set system $\mathcal{M}_1 \cap \mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_1 \cap \mathcal{I}_2)$.

- Optimizing a modular function over $\mathcal{M}_1 \cap \mathcal{M}_2$ is equivalent to optimizing a linear function over $\text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \}$.
- As it turns out, this is a solvable polytope.

Theorem

\[ \mathcal{P}(\mathcal{M}_1) \cap \mathcal{P}(\mathcal{M}_2) = \text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \} \]

- One direction is obvious:
  \[ \mathcal{P}(\mathcal{M}_1) \cap \mathcal{P}(\mathcal{M}_2) \supseteq \text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \} \]
- The other direction is not so obvious
The Matroid Intersection Polytope

Matroid Intersection

Given matroids $M_1 = (X, I_1)$ and $M_2 = (X, I_2)$ on the same ground set, we define the set system $M_1 \cap M_2 = (X, I_1 \cap I_2)$.

- Optimizing a modular function over $M_1 \cap M_2$ is equivalent to optimizing a linear function over $\text{convexhull} \{ x_I : I \in I_1 \cap I_2 \}$.
- As it turns out, this is a solvable polytope.

Theorem

$P(M_1) \cap P(M_2) = \text{convexhull} \{ x_I : I \in I_1 \cap I_2 \}$

- One direction is obvious:
  $P(M_1) \cap P(M_2) \supseteq \text{convexhull} \{ x_I : I \in I_1 \cap I_2 \}$
- The other direction is not so obvious
  - It is conceivable that $P(M_1) \cap P(M_2)$ has fractional vertices
Matroid Intersection

Given matroids $\mathcal{M}_1 = (\mathcal{X}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_2)$ on the same ground set, we define the set system $\mathcal{M}_1 \cap \mathcal{M}_2 = (\mathcal{X}, \mathcal{I}_1 \cap \mathcal{I}_2)$.

- Optimizing a modular function over $\mathcal{M}_1 \cap \mathcal{M}_2$ is equivalent to optimizing a linear function over $\text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \}$.
- As it turns out, this is a solvable polytope.

Theorem

$\mathcal{P}(\mathcal{M}_1) \cap \mathcal{P}(\mathcal{M}_2) = \text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \}$

- One direction is obvious:
  $\mathcal{P}(\mathcal{M}_1) \cap \mathcal{P}(\mathcal{M}_2) \supseteq \text{convexhull} \{ x_I : I \in \mathcal{I}_1 \cap \mathcal{I}_2 \}$
- The other direction is not so obvious
  - It is conceivable that $\mathcal{P}(\mathcal{M}_1) \cap \mathcal{P}(\mathcal{M}_2)$ has fractional vertices
  - Nevertheless, it is true but hard to prove, so we will skip it.
Optimization over Matroid Intersections

Optimization over Matroid Intersection $\mathcal{M}_1 \cap \mathcal{M}_2$

maximize $\sum_{i \in X} w_i x_i$

subject to

$\sum_{i \in S} x_i \leq \text{rank}_{\mathcal{M}_1}(S)$, for $S \subseteq X$.

$\sum_{i \in S} x_i \leq \text{rank}_{\mathcal{M}_2}(S)$, for $S \subseteq X$.

$x_i \geq 0$, for $i \in X$. 

Theorem

Given independence oracles to both matroids $\mathcal{M}_1$ and $\mathcal{M}_2$, there is an algorithm for finding the maximum weight set in $\mathcal{M}_1 \cap \mathcal{M}_2$ which runs in $\text{poly}(n)$ time.

Proof: Using equivalence of separation and optimization, and the fact that all coefficients in the LP have $\text{poly}(n)$ bits.
Optimization over Matroid Intersections

Optimization over Matroid Intersection $\mathcal{M}_1 \cap \mathcal{M}_2$

maximize $\sum_{i \in X} w_i x_i$

subject to

$\sum_{i \in S} x_i \leq \text{rank}_{\mathcal{M}_1}(S)$, for $S \subseteq X$.

$\sum_{i \in S} x_i \leq \text{rank}_{\mathcal{M}_2}(S)$, for $S \subseteq X$.

$x_i \geq 0$, for $i \in X$.

Theorem

Given independence oracles to both matroids $\mathcal{M}_1$ and $\mathcal{M}_2$, there is an algorithm for finding the maximum weight set in $\mathcal{M}_1 \cap \mathcal{M}_2$ which runs in $\text{poly}(n)$ time.
Optimization over Matroid Intersections

**Theorem**

Given independence oracles to both matroids \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \), there is an algorithm for finding the maximum weight set in \( \mathcal{M}_1 \cap \mathcal{M}_2 \) which runs in \( \text{poly}(n) \) time.

Proof: Using equivalence of separation and optimization, and the fact that all coefficients in the LP have \( \text{poly}(n) \) bits.
NP-hardness of 3-way Matroid Intersection

By a reduction from Hamiltonian Path in directed graphs