5.1 Review From Last Lecture

We’ll go over what we did in the last lecture with regard to subset systems and greedy algorithms.

5.1.1 Subset Systems

A subset system \((E, \mathcal{I})\) consists of

- \(E\): A finite set of elements
- \(\mathcal{I}\): A collection of independent subsets of \(E\) that is closed under inclusion.

Note: We can think of the independent subsets as “valid” subsets of \(E\) that solve the optimization problem that we are considering.

Closed under inclusion means that given a set \(i \in \mathcal{I}\), all subsets of \(i\) are also in \(\mathcal{I}\). Here are some examples to clarify these concepts.

5.1.1.1 Example 1

\(E\): \(\{e_1, e_2, e_3\}\)
\(\mathcal{I}\): \(\{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \emptyset\}\)

First, all of the elements of the subsets in \(\mathcal{I}\) are in \(E\). Now we need to verify that \((E, \mathcal{I})\) is closed under inclusion. Consider the first set in \(\mathcal{I}\), \(\{e_1, e_2\}\). All of its subsets are also in \(\mathcal{I}\), namely \(\{e_1, e_2\}, \{e_1\}, \{e_2\}\), and \(\emptyset\). Likewise, all subsets of each of the other sets in \(\mathcal{I}\) are also in \(\mathcal{I}\).

5.1.1.2 Example 2

\(E\): the edges of a graph
\(\mathcal{I}\): the acyclic subsets of the edges (forests)

It is clear that the subsets in \(\mathcal{I}\) are made up of elements from \(E\). Since removing an edge from any forest will result in another forest, we see that this subset system is closed under inclusion.

5.1.1.3 Example 3: Matchings

\(E\): the edges of a graph
\(\mathcal{I}\): subsets of the edges such that no two edges share a vertex
Figure 5.1: Two different matchings on the same graph

Figure 5.1 gives two examples of matchings on the same graph. In each case, the bold edges represent the edges that belong to the matching. The verification that a matching is a subset system is very similar to the previous example. If we remove an edge from any matching, the result is also a matching.

5.1.2 Optimization Problems on Subset Systems

We can define a generic optimization problem for subset systems.

Input: \((E, I), w :\to \mathbb{R}^+\)

Output: the element of \(I\) with maximum weight (that is, a valid subset of \(E\) with maximum weight)

The generic greedy algorithm for optimization problems is as follows:

\[
i = \emptyset \\
\text{sort the elements of } E \text{ by non-decreasing weight} \\
\text{for each } e \in E \\
\quad \text{if } i + e \in I \text{then } i = i + e \\
\text{output } i \\
\text{end for each}
\]

For Example 2 (above), the Maximum Weight Forest (MWF) problem, the greedy algorithm provides the optimal solution.

For Example 3 (above), the Maximum Weight Matching problem, the greedy algorithm does not provide the optimal solution.

In order to characterize the subset systems for which the greedy algorithm provides an optimum solution as well as those for which it does not, we use matroids.

5.2 Matroids

A matroid is a subset system which satisfies the exchange property.
The exchange property:

If \( i, i' \in \mathcal{I} \) such that \(|i| < |i'|\) then
\[ \exists e \in i' - i \text{ such that } i + e \in \mathcal{I} \]

In other words, given two subsets from \( \mathcal{I} \), where one of the subsets has a larger number of elements than the other, there will be some element that is in the larger set and not in the smaller which can be added to the smaller to produce another valid subset.

5.2.1 Examples

From Example 1 above:

\[
E: \{e_1, e_2, e_3\} \\
\mathcal{I}: \{\{e_1,e_2\},\{e_2,e_3\},\{e_1\},\{e_2\},\{e_3\},\emptyset\}
\]

Choose \( \{e_1\} \) as \( i \) and \( \{e_2,e_3\} \) as \( i' \). We can add \( e_2 \) to \( i \) and produce a valid subset. Therefore this subset system satisfies the exchange property and is a matroid. Note that the exchange property only states that we must be able to add some element from \( i' \) to \( i \), not any element.

From Example 3 above:

Let the matching in Figure 5.1(a) be \( i \) and the matching in Figure 5.1(b) be \( i' \). Adding any edge to (a) from (b) that is not already in (a) results in an invalid matching. Therefore this subset system does not satisfy the exchange property and we know that matchings are not matroids.

5.2.2 The Maximum Weight Forest is a Matroid

Theorem 5.1 The Cardinality Theorem

A subset system \((E, \mathcal{I})\) is a matroid if and only if \( \forall A \subseteq E \) if \( i \) and \( i' \) are maximal independent subsets of \( A \) then \( |i| = |i'| \).

We can now use the Cardinality Theorem to demonstrate that the Maximum Weight Forest subset system is a matroid.

From the graph \( E \), choose \( A \), an arbitrary subset of the edges in \( E \). Let \( i \) be a maximal independent subset of \( A \), that is, a maximal acyclic subset of \( A \). The number of edges in \( i \) can be calculated by subtracting the number of connected components in \( A \) from the number of vertices in \( A \). If \( i' \) is also a maximal independent subset of \( A \) the number of edges in \( i' \) can be calculated in the same manner. Because both the number of connected components in \( A \) and the number of vertices in \( A \) are the same for both \( i \) and \( i' \), the number of edges in each is also the same. Thus, by the Cardinality Theorem, the Maximum Weight Forest is a matroid. Figure 5.2 provides an illustration.
5.2.3 Proof that the Greedy Algorithm Solves the Optimization Problem iff the Subset System is a Matroid

**Theorem 5.2** The Greedy Algorithm solves the optimization problem for \((E, \mathcal{I})\) if and only if \((E, \mathcal{I})\) is a matroid.

**Proof:**

(\(\Leftarrow\)): If \((E, \mathcal{I})\) is a matroid, then the Greedy Algorithm produces the optimal solution.

Assume that \((E, \mathcal{I})\) is a matroid, but that the Greedy Algorithm does **not** find the optimal solution.

Let the greedy solution be

\[ i = \{e_1, e_2, \ldots, e_k\} \]

and the optimal solution be

\[ j = \{e_1', e_2', \ldots, e_{k'}\} \]

In fact, \(k\) is equal to \(k'\). By nature, the Greedy Algorithm will produce the maximal solution. Also, the optimal solution must be the maximal solution. Therefore, by the Cardinality Theorem, \(|i| = |j|\), and therefore \(k = k'\).

We assume an ordering of the elements: namely,

\[ w(e_1) \geq w(e_2) \geq \ldots \geq w(e_k) \]

and

\[ w(e_1') \geq w(e_2') \geq \ldots \geq w(e_{k'}) \]

Since \(w(j) > w(i)\), there exists some element \(e_s \in i\) such that \(w(e_s) < w(e_s')\). If this occurs multiple times, then let \(e_s\) be the first such occurrence.

Let

\[ \alpha = \{e_1, e_2, \ldots, e_{s-1}\} \]
and
\[ \beta = \{ e'_1, e'_2, \ldots, e'_{s-1}, e'_s \} \]

Then, by the Exchange Property, \( \exists e'_i \in \beta - \alpha \) such that \( \alpha \cup \{ e'_i \} \in \mathcal{I} \). In other words, there exists an element of \( \beta \) that can be added to \( \alpha \) to produce a set in \( \mathcal{I} \).

It is clear that \( e'_i \in \beta \)

Because of the ordering of \( j \), \( w(e'_i) \) is at least as large as \( w(e'_s) \), and \( w(e'_i) > w(e_s) \). Therefore, \( w(e'_i) > w(e_s) \).

However, if \( i \) is truly the greedy solution, then the Greedy Algorithm should have considered \( e'_i \) before \( e_s \). This is a contradiction.

\((\Rightarrow)\): If the Greedy Algorithm gives the optimal solution, then \((E, \mathcal{I})\) is a matroid.

Assume that \((E, \mathcal{I})\) is not a matroid. We will show that, for a suitable weight function, the greedy algorithm fails.

Because \((E, \mathcal{I})\) is not a matroid there exist \( i, i' \in \mathcal{I} \) such that \( |i| < |i'| \) and \( \neg \exists e \in i' - i \) such that \( i + e \in \mathcal{I} \).

Let \( m = |i| \). Here we present a weight function for which the greedy algorithm fails:

\[
w(e) = \begin{cases} 
m + 2 & \text{if } e \in i \\
m + 1 & \text{if } e \in i' - i \\
0 & \text{otherwise} \end{cases}
\]

With this weight function, the Greedy algorithm produces a solution equal to the set \( i \). Therefore, \( w(i) = m(m + 2) = m^2 + 2m \). The optimal solution contains all of \( i' \), which contains at least \( m + 1 \) elements. \( w(i') \geq (m + 1)^2 = m^2 + 2m + 1 \).

Therefore, \( w(i) < w(i') \), so the greedy algorithm failed to find an optimal solution.

\[ \blacksquare \]

5.2.3.1 Example 4

**Input:** A weighted directed graph \( G = (V, E) \), \( w : E \to \mathbb{R}^+ \).

**Output:** The maximal weight subset of \( E \) such that no two edges point to the same node.

![Graph Example](image)

Figure 5.3: (a) Example of a weighted directed graph. (b) Optimal solution to the graph in (a).

Prove that, in general, the Greedy Algorithm does work on this problem.

\[ E: \text{ edges of the graph.} \]
\[ \mathcal{I}: \text{ subsets of edges in which no more than one edge points to a vertex.} \]
For any subset \( A \in E \), the number of edges in the \textit{maximal} valid subset of \( A \) is the number of vertices in \( A \) with at least one incoming edge.

This satisfies the conditions of the Cardinality Theorem. Therefore, this is a matroid, and the Greedy Algorithm will find the optimal solution.

\textbf{5.2.3.2 Example 5}

\textbf{Input:} \( n \times n \) matrix \( A \), with real numbered entries. Weight for each column of \( A \).

\textbf{Output:} Maximum weight set of linearly independent columns. (Linearly independent: for a set of columns, no column of the set can be produced by a linear combination of the other columns in the set.)

\[
A = \begin{bmatrix}
3 & 1 & 3 & 0 & 2 & 1 & 2 \\
0 & 2 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 2 & 0 & 1 & 0 & 0 \\
2 & 0 & 0 & -1 & 0 & 2 & 3 \\
e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7
\end{bmatrix}
\]

The set \( \{e_1, e_2, e_3, e_4\} \) is linearly independent.
\( \{e_1, e_4, e_5, e_6\} \) is not linearly independent, because \( e_1 - e_5 = e_6 \).
\( \{e_4, e_6, e_7\} \) is not linearly independent, because \( e_4 + 2e_6 = e_7 \).

\textbf{Subset system:}

\( E \): columns of \( A \).

\( I \): subsets of columns that are linearly independent.

It is a fact (from linear algebra) that all maximal linearly independent subsets of a set of vectors have the same cardinality. Therefore, this is a matroid.

\textbf{5.3 The Bipartite Matching Problem}

\textbf{Input:} A bipartite graph \( B = (U, V, E) \). \( U \) and \( V \) are distinct sets of vertices, and \( E \) a set of edges satisfying the property that each edge spans from a vertex in \( U \) to a vertex in \( V \).

\textbf{Output:} A matching (no two edges share a common vertex) of maximum size.

\[
\begin{array}{ccc}
& \bullet & \\
\bullet & \bullet & \bullet \\
\bullet & \bullet & \\
U & \bullet & V \\
\end{array}
\]

\textbf{Figure 5.4: Example of a Bipartite Graph}

Why are we studying this problem?

\begin{itemize}
  \item The techniques we use will apply to a wide class of problems. This problem is not actually a matroid, but we can define the problem \textit{in terms of} matroids.
  \item Also, the problem is interesting in its own right.
\end{itemize}
5.3.1 Application of the Bipartite Matching Problem

We have a group of people that we would like to assign to a number of tasks.
Restrictions: Not everyone can do every task.
   Each person can only do one task.
   Each task only requires one person.

Goal: Maximize the number of tasks performed.

This is equivalent to the bipartite matching problem. $U$ corresponds to individual people, $V$ to the tasks, and $E$ to what task a person can perform.

5.3.2 Solution to the Bipartite Matching Problem

\[
\begin{array}{c}
  \bullet \\
  \bullet \\
  U \\
  \bullet \\
  \bullet \\
  V \\
\end{array}
\]

Figure 5.5: The Bipartite Matching Problem is not a matroid

As stated, the Bipartite Matching Problem is not a matroid. For example, consider Figure 5.4. Let $i$ be the set of bold edges, and $i'$ the set of thin edges. There is no thin edge that can be added to the set of bold edges that results in a matching.

Although this problem is not a matroid, we can describe it in terms of matroids.

First, treat $U$ independently of $V$. Look at the problem of finding a maximal set of edges such that at most one edge is incident to any node of $U$. This problem is a matroid (and is is similar to the directed graph problem). Call this matroid $(E, \mathcal{I})$.

Now treat $V$ independently of $U$. Find the maximal set of edges such that at most one edge is incident to any node of $V$. This produces a second matroid, $(E', \mathcal{I}')$.

The valid matchings correspond to the intersection of the two matroids, i.e. they are the subsets of edges in $\mathcal{I} \cap \mathcal{I}'$. 