Lecture 21: Discrete optimization, submodular functions

Today and for the next few lectures we will see how to use the techniques from convex optimization to obtain algorithms for problems that are of a very different nature: the problems we will consider fall under the umbrella of discrete (or combinatorial) optimization.

Discrete Optimization

In discrete optimization, we have a finite collection V of items / elements that we call the ground set.

We also have a set function $f: 2^V \to \mathbb{R}$ that assigns a real value to each subset $S \subseteq V$ of the ground set (we can think of $f(S)$ as the cost or the value of the set S of items).

Finally, we have a collection $\mathcal{F} \subseteq 2^V$ of feasible sets.

The goal is to select a feasible set $S \in \mathcal{F}$ of minimum or maximum value $f(S)$:

$$\boxed{\begin{array}{ll} \min/\max & f(S) \\ \text{subject to} & S \in \mathcal{F} \end{array}}$$

This is a very general framework and many discrete optimization problems fit into this framework. Indeed, one can think about all the problems that one encounters in algorithms: problems such as shortest paths, matchings, maximum flows and minimum cuts, assignment problems, etc., are all discrete optimization problems.

As we have seen in algorithms class, not all combinatorial problems are alike: some problems are solvable in polynomial time, others are NP-hard or worse.

What makes some of these problems so different? One answer lies in the underlying structure of the problem, and it is very useful to identify and study general structures that may lead to algorithms.

We can also think about it as follows. It is not hard to see that the discrete optimization framework is too general to allow for any algorithms. If f and $\mathcal{F}$ can be arbitrary, there is nothing we can do from an algorithmic point of view.

Thus an important question is: which properties of f and $\mathcal{F}$ make the optimization problem tractable?

One such property that arises in many settings is that of submodularity.

Submodular Optimization

Let $f : 2^V \to \mathbb{R}$ be a set function. We say that f is submodular if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad \text{for all sets } A, B \subseteq V$$

An equivalent definition of submodularity is the following.

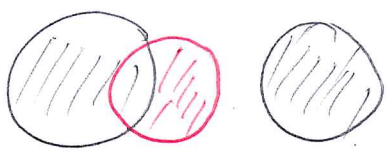
$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \quad \text{for all } A \subseteq B \subseteq V \text{ and all } e \in V \setminus B.$$

The second definition highlights a remarkable property of submodular functions: they have "diminishing returns": the quantity $f(A \cup \{e\}) - f(A)$ is called the marginal gain of adding the element e to the set A; the second definition tells us that the marginal gain can only decrease as the set grows.

Submodular functions arise in many fields, from Economics to Mathematics and Computer Science, and they play a central role in Discrete Optimization. Many functions arising in applications are submodular. Let us see only a couple of examples of submodular functions.

Example I: Coverage functions.

Let us start with the following example. Suppose our ground set is a collection of disks in the plane. If S is a subset of these disks, let $f(S)$ be the total area covered by the disks in S.
Then we can easily see that f is submodular, since it clearly has diminishing returns: the more disks we have, the less new area will any new disk cover.



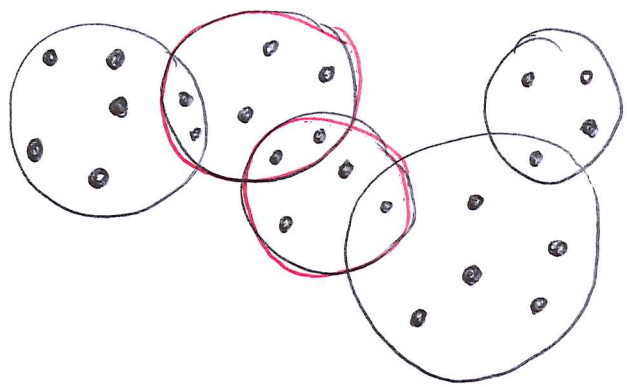
We can make this precise and formally verify that the second definition of submodularity applies.

More generally, we can consider a collection $\mathcal{C}$ of sets from a ground set N. For simplicity, let us assume N is finite. Each set $C \in \mathcal{C}$ is a subset of N. We can associate a set function f with this set system, called the coverage function. The ground set of f is the sets of $\mathcal{C}$. If we number the sets of $\mathcal{C}$ as $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ then we can write the ground set of f as $\{1, 2, \dots, m\}$: $f : 2^{\{1,2,\dots,m\}} \to \mathbb{R}$.

The value of a set $S \subseteq \{1, 2, \dots, m\}$ is the total number of distinct elements of N that appear in the sets $\{C_i : i \in S\}$:

$$f(S) = \left| \bigcup_{i \in S} C_i \right|$$

For example, N can be points in the plane, and $\mathcal{C}$ is a collection of points like this one:



Then the two sets highlighted in red above together they contain 10 different points, thus their coverage is 10.

It is straightforward to verify that the coverage function of any set system is submodular (we can easily check that the diminishing returns property is satisfied).

<u>Example II</u> : Cut functions.

Another example of a submodular function is the cut function of a graph.

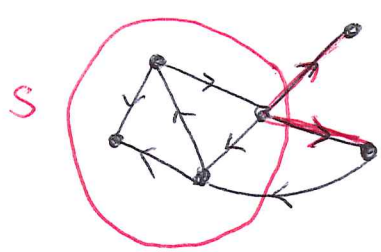
Consider a graph $G=(V,E)$ (it can be directed or undirected).

The ground set of the cut function of G is the set of vertices of G:

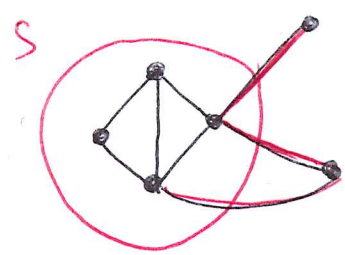
$$f : 2^V \to \mathbb{R}.$$

The value of a set $S \subseteq V$ of vertices is the total number of edges leaving S, that is, edges whose tail is in S and its head is not in S (if G is undirected, the edge is leaving S if it has an endpoint in S and an endpoint not in S).

For example:



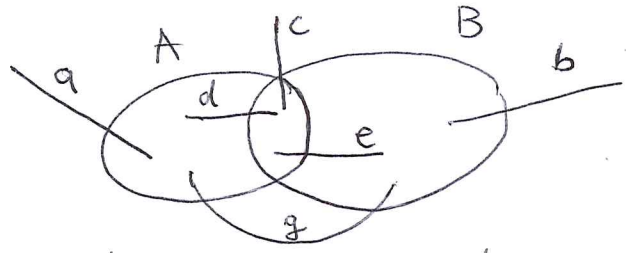
$f(S) = 2$



$f(S) = 3$

It is less obvious that the cut function is submodular, but we can show that by verifying either definition.

We can verify the first definition as follows. Consider two sets A, B of vertices



We can see the different types of edges in the figure above (we assume the graph is undirected, but essentialy the same argument will work for directed). The label indicates how many edges of each type, for example, we let a denote the number of edges with an endpoint in $A \setminus B$ and the other in $V \setminus (A \cup B)$

We have:

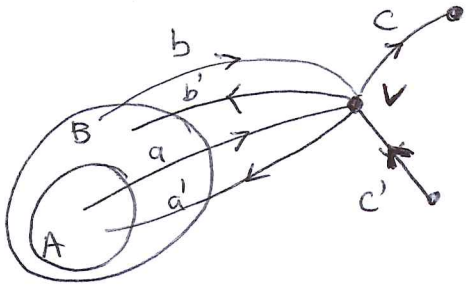
$$f(A) = a + c + e + g \qquad f(A \cap B) = c + d + e$$

$$f(B) = b + c + d + g \qquad f(A \cup B) = a + b + c$$

Thus
$$\begin{aligned} f(A) + f(B) &= a + b + 2c + d + e + 2g \\ &\geq a + b + 2c + d + e \\ &= f(A \cap B) + f(A \cup B) \end{aligned}$$

Let us also verify the second definition, and this time let us consider directed graphs.

Consider two sets of vertices $A \subseteq B$ and a vertex $v \notin B$.



We can see the different types of arcs incident to v in the figure above.

We have:

$$f(A \cup \{v\}) - f(A) = c + b' - a$$

$$f(B \cup \{v\}) - f(B) = c - (a+b)$$

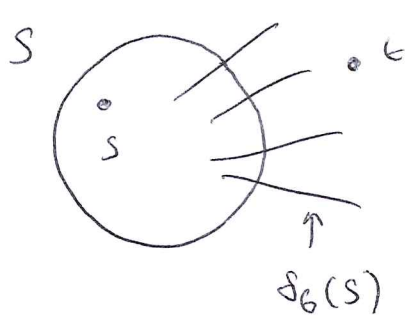
Thus clearly $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$.

Submodular optimization problems are discrete optimization problems where the set function f is submodular.

There are two types of submodular optimization problems: submodular minimization problems and submodular minimization problems. These two classes of problems are quite different from an algorithmic point of view. We will not cover submodular maximization in this class, but we can see the contrast between submodular minimization and maximization via the following example.

We saw that the cut function of a graph is submodular. This observation allows us to see that two graph problems, minimum cut and maximum cut, are submodular optimization problems.

<u>Minimum s-t cut</u>: Let $G = (V, E)$ be a graph, and let s, t be two vertices. The minimum s-t cut problem asks us to find a cut separating s and t with a smallest number of edges leaving. More precisely, we want to find a set S with $s \in S$ and $t \notin S$ such that $|\delta_G(S)|$ is minimized.



This problem feels like it is asking us to minimize the cut function of G: $\min_{S \subseteq V} |\delta_G(S)|$. This is almost true, but we have the additional requirement that $s \in S$ and $t \notin S$. We can get around this as follows.

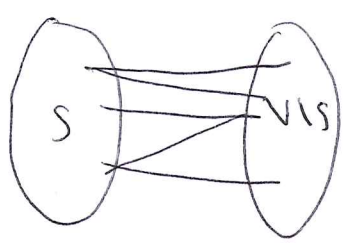
Let $f : 2^{V \setminus \{s,t\}} \to \mathbb{R}$ be the function such that, for all $S \subseteq V \setminus \{s,t\}$, we have $f(S) = |\delta_G(S \cup \{s\})|$.

Then we can easily show that f is submodular like we did for the cut function and clearly the min cut problem asks us to find $S^* \cup \{s\}$ where

$$S^* \in \arg\min_{S \subseteq V \setminus \{s,t\}} f(S).$$

Therefore the minimum cut problem is an <u>unconstrained</u> submodular minimization problem.

<u>Maximum cut</u>: Let $G = (V, E)$ be a graph. The maximum cut problem asks us to find a partition $(S, V \setminus S)$ of the vertices with a maximum number of edges from S to $V \setminus S$.



This problem is precisely $\max_{S \subseteq V} f(S)$ where f is the cut function of G, and thus it is an unconstrained submodular maximization problem.

You may have encountered those problems in your algorithms class. If so, you may already know that they are very different from an algorithmic point of view: the minimum cut problem is solvable in polynomial time, but the maximum cut problem is NP-hard (but it can be approximated within a constant factor, $\frac{1}{2}$ is easy)

A similar phenomenon happens in the submodular setting:

* unconstrained submodular minimization is solvable in polynomial time
* unconstrained submodular maximization is NP-hard, but we can find a constant factor approximation in polynomial time (a $\frac{1}{2}$ approximation is not easy, but it is possible)

The landscape is quite different for <u>constrained</u> submodular optimization problems. Even though we can solve the unconstrained submodular minimization problem in polynomial time, even very simple constraints such as a cardinality constraint make the problem very hard to approximate, with typical hardness results overruling an approximation better than $\Theta(\sqrt{n})$.

Constrained submodular maximization problems tend to be much more tractable, and good (often constant factor) approximations are known for natural constraints such as cardinality, matroid, knapsack, etc.