COMP/MATH 552 Lecture 1: Course overview*

Yaqiao Li

January 8, 2018

Combinatorial optimization involves choosing an element from a finite set which maximizes or minimizes a given objective function. If the problem is completely unstructured then we would need to be given the input as a list of the elements of the set, with the corresponding function values. In this case, the problem is easy to solve efficiently: while reading the input, we simple keep track of the elements with the highest and lowest function values found so far and output these elements at the end of our scan.

Typically, however, the finite set is given in a more compact form using a graph or other combinatorial object. In this case, we are interested in algorithms which run much more quickly than one which generates every candidate *feasible solution* and churns through them in a brute force fashion to find the optimal one.

One simple example is the minimal spanning tree problem: given a graph G whose edges have been assigned non-negative weights, find a spanning tree minimizing the sum of the weights of its edges. Now, a graph on n vertices can have as many as n^{n-2} spanning trees, and we certainly don't want to churn through all of these. However, it turns out that we can solve this problem by being greedy: simply order the edges so that their weights are non-decreasing, and scan through them, adding each edge to the current partial tree as long as it does not create a cycle. Rather than meandering through all the trees, we march directly to the one that interests us.

A second more complicated example is finding an independent set[1] in a graph G with the maximum possible number of vertices. Again, a graph on n vertices can have as many as 2^n indepenent sets, and we certainly don't want to churn through all of these. It turns out that for certain specially structured graphs, such as bipartite graphs[2], we can exploit the structure of the graph to get an efficient algorithm for finding a maximum independent set. On the other hand, for other less well-structured graphs, this is impossible.

Our focus on this course will be on understanding what type of structural properties allow us to develop efficient algorithms for optimization problems. The theme is that there are some specific types of structures which reoccur again and again in individual problems. Our focus will not be on solving practical problems but on setting out what these special structures are, and seeing a variety of examples of the occurrence of each.

We briefly survey the type of special structures which will interest us.

*The lecturer is grateful to Bruce Reed for many valuable discussions and guidance in forming this overview.

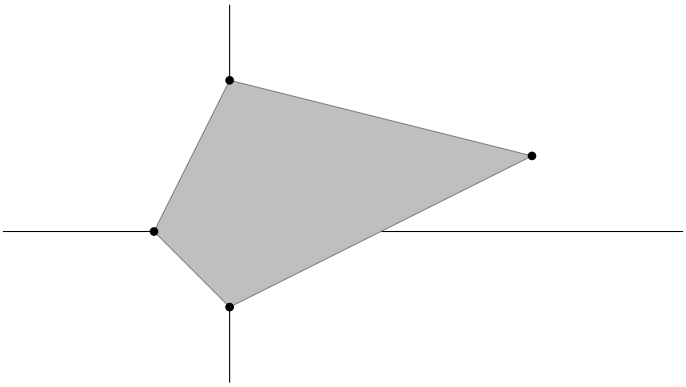
[1] An independent set is a set of vertices no two of which are joined by an edge.

[2] Recall that a graph is bipartite if its vertex set can be partitioned into two independent sets.

1 Linear programming and polytopes : from discrete to continuous

A polytope (Figure 1) is the intersection of some set of half spaces in $\mathbb{R}^n$.

Figure 1: A polytope in $\mathbb{R}^2$

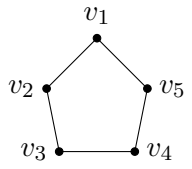


A Linear Programming problem involves maximizing or minimizing a linear objective function over a polytope. Equivalently it involves finding the point in the polytope which is furthest in some direction. An Integer Programming problem involves maximizing or minimizing a linear objective function over the points in the polytope which have integer coordinates. The LP relaxation of an IP is the LP with the same objective function and polytope.

All of the combinatorial optimizations problems that we discuss can be formulated naturally as Integer Programming problems[3]. We will see that for those that are specially structured, this allows us to use Linear Programming techniques to solve them. In general however, solving an LP relaxation only gives us a bound on the optimal solution value.

For example, consider the maximizing independent set problem on C_5.

Figure 2: C_5



[3]Indeed since Integer Programming is NP complete, any problem which has a chance of being efficiently solved can be formulated as an IP, usually naturally.

The optimization problem can be formulated as

$$\begin{aligned} & \max \sum_{i=1}^{5} x_{v_i} \\ \text{s.t.} \quad & x_{v_i} + x_{v_{i+1}} \leq 1, \quad i = 1, ..., 5; \text{ (where } x_{v_6} := x_{v_1}) \\ & x_{v_i} \geq 0, \quad i = 1, ..., 5; \\ & x_{v_i} \in \mathbb{Z}, \quad i = 1, ..., 5. \end{aligned} \tag{1}$$

Its corresponding LP relaxation is

$$\begin{aligned} & \max \sum_{i=1}^{5} x_{v_i} \\ \text{s.t.} \quad & x_{v_i} + x_{v_{i+1}} \leq 1, \quad i = 1, ..., 5; \text{ (where } x_{v_6} := x_{v_1}) \\ & x_{v_i} \geq 0, \quad i = 1, ..., 5. \end{aligned} \tag{2}$$

Let us use OPT_{IP} and OPT_{LP} to denote the optimal values of the IP in (1) and LP in (2), respectively. Obviously $OPT_{IP} = 2 < OPT_{LP} = 5/2$. The reason for this is that the polytope in question has a non-integer vertex $(1/2, 1/2, 1/2, 1/2, 1/2)$. It is not hard to see that $OPT_{IP} = OPT_{LP}$ if all the vertices of the polytope are integral. This naturally leads us to study *integer polytopes*.

Definition 1. *A polytope $P \subseteq \mathbb{R}^n$ is called an integer polytope if all its vertices are integer vectors in $\mathbb{R}^n$.*

One of our focuses will be combinatorial optimization problems whose natural IP formulation has LP relaxation whose underlying polytope is integral. Consider, for example, the problem of finding a maximum independent set in a graph. Letting $\mathcal{C}$ be the set of cliques of G, the LP relaxation involves maximizing over the following polytope:

$$\begin{cases} \sum_{v \in C} x_v \leq 1, & \forall\ C \in \mathcal{C}; \\ x_v \geq 0, & \forall\ v \in V. \end{cases} \tag{3}$$

It is not difficult to see that the existence of odd cycles such as the C_5 we discussed above, prevents this polytope from being integral. It turns out that for the bipartite graphs, which have no such odd cycles, this polytope is integral. Furthermore, this polytope can be described using a small number of inequalities because in a bipartite graph the only cliques are edges and vertices. As we shall see this provides us with a simple efficient algorithm to find a maximum independent set in a bipartite graph.

In a non-bipartite graph we have two problems. The first is that the polytope may not be integral. The second is that the number of cliques may be exponential in the size of the graph, so that writing down a list of inequalities for the polytope may take too long.

We deal with the first problem as follows.

Definition 2. *Given a finite set $S \subseteq \mathbb{R}^n$, its convex hull P is defined as*

$$P := \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{k} \lambda_i x_i \quad \textit{for some } k, \textit{such that } 0 \leq \lambda_i \leq 1, \sum_{i=1}^{k} \lambda_i = 1, \textit{and } x_i \in S\}.$$

Given any combinatorial optimization problem whose solutions are integer valued vectors, by taking the convex hull of these vectors we create an integer polytope such that to solve a combinatorial optimization problem with a linear objective function, we need just solve the corresponding LP over the polytope.

At first sight it may seem strange and even stupid to do such a continuous extension. Even though the LP offers us a lower or upper bound for the original problem, now on solving the LP, it seems we are facing an even harder problem as the set of feasible solutions become much larger. However, the beauty and usefulness of leaving from a discrete and finite world into a continuous space lie in the following:

- many analytical tools can be used in a continuous space but not a discrete world;
- the optimal value of the LP is not only a lower or upper bound, but also equals to the optimal value of the original problem;
- there are polynomial time algorithms for solving ALL LP problems, and there is a simple algorithm solving practical LP efficiently;
- there is a nice duality theory for LP that helps us understand and sometimes even design a good algorithm for the original optimization problem;
- it offers geometric insights.

Let us elaborate on some of these points. The first point above has many analogies in other parts of mathematics and computer science. In number theory, one goes from elementary number theory full of clever individual techniques to analytic number theory where many problems can be treated systematically. Some most important number theory problems naturally live in continuous spaces (e.g. Riemann hypothesis) and at the same time have tremendous implication in the discrete world. In geometry, one goes from Euclidean axiomatic geometry (where even to show the three altitudes of a triangle intersect at one point needs some work) to analytic (Cartesian) geometry. Upon doing so, suddenly many planar geometric problems reduce more or less to trivial mechanical calculation. In complexity theory, a recent development is the continuation of communication complexity to information complexity. Instead of considering the number of bits transmitted in a system, which is a naturally discrete object, one considers the information[4] revealed in the system. Doing so leads into a continuous world, and much can be gained from this continuation process. In this context, the relaxation of discrete optimization problems into a continuous world is then just natural, and is actively pursued in current research.

The second point comes from the fact that the objective functions we are optimizing are linear[5] and the specific continuous extension we chose: the convex hull, so the polytope is a convex body in nature. Basic analysis shows optimizing linear functions over a convex domain is usually much easier optimizing them over other structures.

As set out in the third point, the ellipsoid method gives a polynomial time algorithm for ALL LP, and the simplex algorithm runs fast for most practical LP. This gives, in theory, a very satisfactory answer to combinatorial optimization problems with linear objective functions, if we know a small system of inequalities describing the polytope that we need to optimize over. At least in theory,

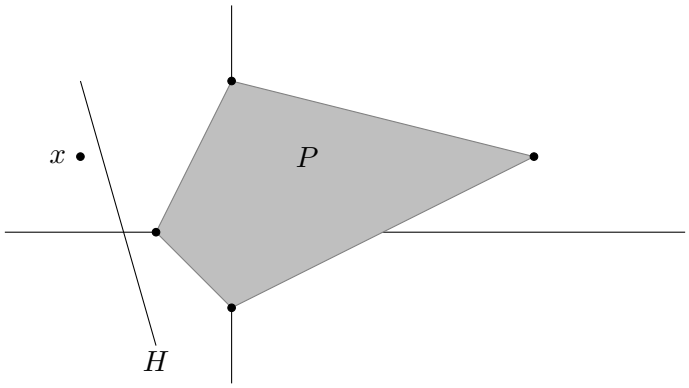
[4] In the sense of Shannon entropy.

[5] Later we will see convex or concave objective functions.

if we are content on knowing there is a polytime algorithm for a problem, then suddenly many combinatorial optimization problems with linear objective functions are solved! There is no need to invent specific technique for each individual combinatorial problem anymore.

It turns out that, as discovered by Grötschel, Lovász, and Schrijver, we can extend the range of the ellipsoid method to certain integer polytopes for which we do not have a short system of defining inequalities. They showed that the optimization of all linear objective functions over a polytope is roughly equivalent to the separation problem. A separation problem is, for a given point x and a polytope P, to answer either $x \in P$ or show a hyperplane H that separates x from P, see Figure 3. This seemingly unrelated problem turns out to be equivalent to the LP optimization problem: that if we have a polytime algorithm to solve one of them, we can solve the other in polytime too!

Figure 3: A separation of x from P by H in $\mathbb{R}^2$



Duality is a general theme and tool in modern mathematics and theoretical computer science. Given an LP in the maximization form,

$$\begin{aligned} &\max w^T x \\ \text{s.t.} \quad &x \in P, \end{aligned} \tag{4}$$

where P is some polytope, the duality theory of LP says, under some conditions, that the optimal value of (4) equals to the optimal value of another minimization problem,

$$\begin{aligned} &\min c^T y \\ \text{s.t.} \quad &y \in P', \end{aligned} \tag{5}$$

for some other polytope P'. Here the polytope P' and c in (5) depend on P and w in (4) in a natural way. On the one hand, this theorem immediately unifies many min-max phenomena in combinatorial optimization (e.g., max flow-min cut, maximal macthing and minimal vertex cover, ...). On the other hand, it suggests a so-called primal-dual algorithm that can sometimes help us find a good algorithm for the original combinatorial optimization problem. We will elaborate this further in this course.

Although LP has polytime algorithms, it is still desirable to find combinatorial algorithms for specific problems (e.g., they are faster). We will see one case of the so-called primal-dual algorithms in the weighted matching problem on general graphs: a beautiful theorem of Edmond. As an

example of the application of matching, we study the optimal T-join problem and its application in the Chinese postman problem.

2 Matroids as a unifying structure

We have the experience that greedy algorithm is quite useful in many occasions, how to tell when it can be applied? Is there any structure behind a problem that makes the greedy algorithm applicable? It turns out there is a simple answer: the matroids.

Definition 3. *Given a finite set X, let $M \subseteq 2^X$ be a family of subsets of X. We call M a matroid if it satisfies:*

- $\emptyset \in M$;
- $A \subseteq B \in M$ *implies* $A \in M$;
- *if* $A, B \in M$ *and* $|A| < |B|$, *then there exists* $x \in B$ *such that* $A \cup \{x\} \in M$.

Given a graph G, let X be the set of its edges, and M be the family of edge sets of forests in G. It can be shown that M is a matroid, call it the *graphical matroid*. In fact, this matroid is exactly the underlying structure behind Kruskal's algorithm for minimal spanning tree. In each step we construct an element of M, i.e., a forest of G, and we do this in a greedy way by extending the current element of M by adding the minimal weight edge which yields a new element of M. We will show that this greedy process works not only for the minimal spanning tree problem (i.e., the graphical matroid), but also for all matroids. And conversely, matroids characterize the structure to which we can successfully apply the greedy algorithm!

Perhaps a bit surprisingly, it turns out there are more things to explore in the matroid structure than the greedy algorithm. As an example we will study:

> **The matroid intersection problem**: given two matroids M and N, both defined on a set X whose elements have weights, find $S \in M \cap N$ that has the maximal weight.

We will study an algorithm due to Edmonds showing this can be done efficiently. It turns out that several classical combinatorial optimization problems can be viewed as special cases of matroid intersection. Below are two examples:

- Consider a bipartite graph G whose vertices are partitioned into two sets U and V. Obviously the vertices in U naturally define a partition of the edges of G. Let M_U be the family of sets of edges in which every set is formed as: for each $v \in U$, you can pick at most one edge incident to it. Let M_V be defined similarly. It is not hard to see that $M_U \cap M_V$ gives exactly the matchings in G. Hence finding a maximal matching in G reduces to the matroid intersection problem!
- For the second example, given a directed graph G, an *arborescence* rooted at a vertex r is a directed tree, as a subgraph of G, whose edges all point away from r. The task is to find an arborescence with maximal weight. This also reduces to the matroid intersection problem as follows. Let M be the graphical matroid of G as we saw before. Let N be a matroid in which each set is formed by choosing, for every vertex $v \in G$, at most one directed edge goes into v. Then our problem reduces to the matroid intersection for $M \cap N$.

3 Submodularity as discretized convexity or/and concavity

The objective function in LP is linear, hence might not be applicable in many optimization problems. A natural extension are the submodular functions. Interestingly, viewing the matroids as a generalization of matrices, they share an important property: the rank of a matroid as well as the rank of a matrix are submodular.

Definition 4. *Given a finite set X, a function $f : 2^X \to \mathbb{R}$ is called submodular if for any $A \subseteq B \subseteq X$ and $x \notin B$,*

$$f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A). \tag{6}$$

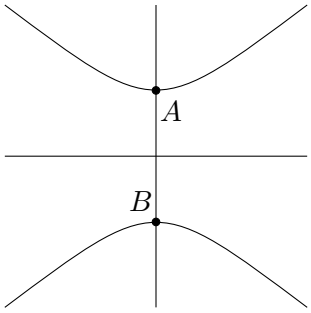
The condition in (6) is sometimes called the *diminishing return*: you get less return if you start from a larger set. This diminishing return property immediately suggests that many functions in economics and algorithmic game theory are submodular.

Obviously the matrix rank is submodular. Some more natural examples are:

- The weight function: let $w : X \to \mathbb{R}$ be a weight on X, define $f(S) = \sum_{x \in S} w(x)$. Obviously f is submodular. Many combinatorial optimization problems are equivalent to find a set S maximizing $f(S)$, perhaps under some constraints.
- The cut function: given a graph G, for any subset S of vertices, define $c(S) := |\delta(S)|$ to be the size of the cut of S, that is, the set of edges between S and $V(G) \backslash S$.
- The entropy function: given a finite set of random variables $X = \{X_1, X_2, \ldots, X_n\}$, for any subset $S \subseteq X$, define $H(S)$ to be the joint entropy of those random variables in S. Then H is submodular, which is a direct consequence of the fact that conditioning reduces entropy: you can learn less new information if you already command the knowledge of a larger set of random variables.

What properties does a submodular function have? This is the central question that we will study in some level of depth. The interesting thing is: submodular functions behave both like concave functions and convex function in the continuous world. Recall from analysis that it is easy to minimize a convex function as well as to maximize a concave function, see in Figure 4.

Figure 4: Convexity and concavity

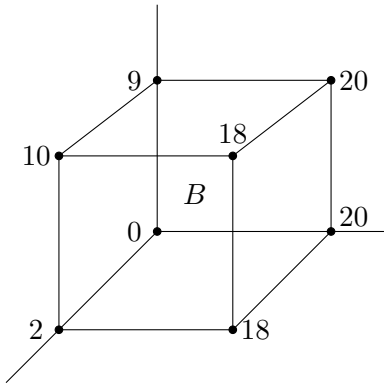


In fact, if we think of the *return*, in the sense of (6), as the discrete analogue for derivatives of differentiable functions, the definition immediately suggests that submodular functions are analogous to concave functions, hence one may expect that it might be easy to maximize submodular

functions. Surprisingly, it turns out the opposite is true: maximizing a submodular function is generally not easy, but minimizing is! This shows the convexity nature of submodular functions. Indeed, Lovász built the exact connection between submodular functions and convexity by using a continuous extension.

Let us give a peek on Lovász's extension. Let f be a submodular function defined on 2^X where $|X| = n$, equivalently, f can be viewed as a boolean function $f : \{0,1\}^n \to \mathbb{R}$ that assigns values to the vertices of the n-dimensional hypercube $\{0,1\}^n$, see Figure 5.

Figure 5: A submodular function on a hypercube B



Now f is a discrete function defined only on finitely many points in $\mathbb{R}^n$, Lovász defined the following extension $g : [0,1]^n \to \mathbb{R}$ of f, as

$$g(x_1, x_2, \ldots, x_n) := \mathbb{E}_{\lambda \sim [0,1]}[f(\{i : x_i \geq \lambda\})].$$

It is not difficult to see that $g = f$ when $x \in \{0,1\}^n$, and in fact one may find an explicit analytical formula for g and can then furthermore extend g to be defined on the whole space $\mathbb{R}^n$. A central theorem of submodular functions due to Lovász says: the discrete function f is submodular if and only if the continuous function g is convex! This allows us to use convex optimization technique (e.g.: ellipsoid algorithm we mentioned before for LP) to solve minimization problems of submodular functions.

Although maximizing a submodular function is generally hard, we will see when the function is monotone, the simple greedy algorithm (again!) gives a $(1 - \frac{1}{e})$-approximation algorithm, and in the general case, how the multilinear extension of submodular functions leads to continuous greedy algorithms.

A submodular function also naturally defines an associated polytope and we will study some properties of it during the course (e.g.: we will show an intersection theorem of two submodular polytopes, generalizing the matroid intersection theorem). Due to its ubiquitous nature, submodular optimization is a quite active research area right now and has many applications in economics, algorithmic game theory, machine learning, approximation algorithms, etc.