CS 466 / 666 Algorithm Design and Analysis, Spring 2020

Lecture 17: Linear Programming

We start the third part of the course and the main object of interest is linear programming.

Today, we introduce linear programming relaxations for combinatorial optimization problems, and explain in what situations they can be used to solve the problem exactly, and the key concept here is "corner" solutions.

We also mention how to solve linear programs efficiently, and how separation implies optimization.

Linear Programs

In a linear program, we have a set of n variables $x_1, \dots, x_n$.

We can add linear constraints to the variables, e.g. $x_1 + 3x_2 - 5x_3 \leq 7$, $x_1 + 4x_2 = 6$, etc (but not strict inequalities).

And then we can optimize a linear objective function such as $\max 3x_1 + 2x_2 - x_3$ or $\min x_1 + 2x_2 - x_3$.

More compactly, if there are m constraints, we can put the constraints as the rows of a matrix $A \in \mathbb{R}^{m \times n}$, the right hand side of the constraints as a vector $b \in \mathbb{R}^m$, and the coefficients of the objective function as a vector $c \in \mathbb{R}^n$, so that the linear program can be written as:

$$\max \langle c, x \rangle \quad Ax \leq b$$

or

$$\max \langle c, x \rangle \quad Ax = b, \; x \geq 0$$

Probably the former form is called the canonical form and the latter form is called the standard form, but we will just refer the former as the inequality form and the latter equational form.

Anyway, they are equivalent and we can rewrite the former form into the latter form and vice versa, e.g. by introducing new variables $s \in \mathbb{R}^m$ and write $Ax \leq b$ as $Ax + s = b$ and $s \geq 0$.

Okay, so this is the definition of a linear program.

Even though we just have linear constraints and objective, LP is very expressive and can model many problems.

Interger linear programs

Many combinatorial optimization problems can be easily formulated as an integer linear program, where we also have the additional constraints $x_i \in \mathbb{Z}$ or just $x_i \in \{0,1\}$.

For example, for bipartite matching, we can create one indicator variable for each edge e, and write the ILP:

$$\max \sum_{e \in E} w_e \cdot x_e$$

$$\sum_{e \in \delta(v)} x_e \leq 1 \quad \text{for each } v \in V, \text{ where } \delta(v) \text{ denotes the set of edges incident on } v$$

$$x_e \in \{0,1\} \quad \text{for each } e \in E.$$

The constraints $x_e \in \{0,1\}$ $\forall e \in E$ say that either we pick e by setting $x_e = 1$ or not by setting $x_e = 0$.

The constraints $\sum_{e \in \delta(v)} x_e \leq 1$ then say that we can pick at most one edge incident on a vertex v.

The objective is to maximize the total weight of the edges picked.

Similarly, we can formulate the maximum independent set problem as an integer linear program.

$$\max \sum_{v \in V} c_v x_v$$

$$x_u + x_v \leq 1 \qquad \text{for each edge } uv \in E$$

$$x_v \in \{0,1\} \qquad \text{for each vertex } v \in V$$

So, that's very convenient. but it just says that solving integer linear program is NP-hard in general.

Linear programming relaxations

As we will discuss later, there are polynomial time algorithms to solve linear programs.

So, one common approach to tackle combinatorial optimization problems is to write a linear program to pretend to be the integer linear program.

The natural way to do this is to write $x_e \in \{0,1\}$ as $0 \leq x_e \leq 1$ and hope for the best.

As you may imagine. this may not work so well, as now we are now optimizing over a much larger domain, the set of solutions in $\mathbb{R}^n$ or $[0,1]^n$, rather than the set of solutions in $\{0,1\}^n$.

Consider the maximum independent set problem in a complete graph and we know the integral optimal solution is one, but the LP solution $x_i = \frac{1}{2}$ $\forall i \in V$ is a feasible solution to the LP solution with objective value $n/2$ assuming all weights are one.

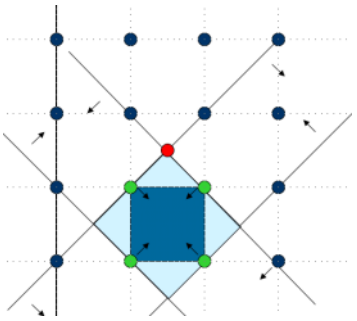
So, the LP always says the graph has an independent set of size $n/2$ and doesn't give useful information.

Perhaps surprisingly, for most of polynomial time solvable combinatorial optimization problems, we can actually extract optimal integral solutions from the LP relaxations.

In fact, it is widely accepted as a unifying algorithmic framework for polytime solvable optimization problems.

Let us first have some intuitions about why this could be the case.

Geometric interpretation



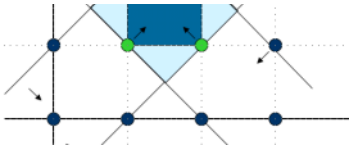
We are working over $\mathbb{R}^n$ ($\mathbb{R}^2$ in the picture)

Each constraint can be thought of as a halfspace.

The set of feasible solution is the intersection of the halfspaces.

We are interested in the integral points in the feasible set.

Optimizing a linear objective function is to find a point furthest in some



We are interested in the integral points in the feasible set.

Optimizing a linear objective function is to find a point furthest in some direction, e.g. if we maximize y, then the top point is the optimal point.

Intuitively, given any direction to optimize, there are always some "corner points" that achieve optimality.

So, if we could show that all the corner points are integral, then we can conclude that the LP relaxation always has integral optimal solutions.

In other words, we need to prove that the linear constraints define precisely the convex hull of the integral solutions.

In the following, we define formally the corner points, and then prove that there are always optimal corner points.

Corner points

Here we make the concept of corner points formally.

Consider an LP of the equational form: max $\langle c, x \rangle$ over the polytope $P := \{ x \mid Ax = b, x \geq 0 \}$, where $A \in \mathbb{R}^{m \times n}$.

We assume without loss of generality that the rows of A are linearly independent.

There are three possible definitions of corner points.

① Vertex solutions: A solution $x \in P$ is an extreme point solution if $\nexists\, y \neq 0$ such that $x+y \in P$ and $x-y \in P$.

This is a geometric interpretion that a corner point is not the average of two feasible points.

② Extreme point solutions: A solution $x \in P$ is an extreme point solution if $\exists\, c \in \mathbb{R}^n$ such that $x \in \mathbb{R}^n$ is the unique optimal solution. This is an optimization interpretion of a corner point.

③ Basic solutions: This definition is algebraic and is easier to deal with when we do calculations.

Let $\text{supp}(x) := \{ i \mid x_i > 0 \}$ be the set of nonzero variables.

Note that x_i corresponds to the i-th column of A.

We say x is a basic solution if the columns correspond to $\text{supp}(x)$ are linearly independent.

In the literature, the names are used interchangably, probably because of the following result.

Proposition The three definitions are equivalent.

Proof We will prove ③ ⇒ ② ⇒ ① ⇒ ③. Let $A \in \mathbb{R}^{m \times n}$ with $m \leq n$ and A is of rank m.

③ ⇒ ②: Let $S = \text{supp}(x)$. By ③, the columns of S are linearly independent.

Since A is of full rank, if $|S| < m$, we can extend S to m linearly independent columns, and we assume without loss of generality that $S = \{1, 2, \dots, m\}$.

Now, consider the objective function $\min \sum_{i=1}^{n} c_i x_i$ where $c_i = 0$ if $i \leq m$ and $c_i = 1$ if $i > m$.

Then, x is a solution with objective value zero, and we claim that it is the unique one

Note that any solution y with objective value zero must have $y_i = 0$ for $i > m$, as $y_i \geq 0$ and $c_i = 1$ for $i > m$.

So, $\text{supp}(y) \subseteq \{1, \ldots, m\}$. But then there is only one solution satisfying $Ax = b$ because the first m columns are linearly independent, and hence we must have $y = x$.

② ⇒ ①: Suppose x is the unique optimal solution for the objective $\min \langle c, x \rangle$.

Assume by contradiction that x is not a vertex solution, i.e. $\exists y \neq 0$ s.t. $x+y \in P$ and $x-y \in P$

Then, either $\langle c, x+y \rangle \leq \langle c, x \rangle$ or $\langle c, x-y \rangle \leq \langle c, x \rangle$ (or both if $\langle c, y \rangle = 0$), contradicting x is the unique optimum.

① ⇒ ③: We prove the contrapositive that if x is not a basic solution, then x is not a vertex solution.

Let $S := \text{supp}(x)$. Since x is not basic, the columns in S are linearly dependent, and $\exists y \neq 0$ s.t. $\text{supp}(y) \subseteq S$ and $Ay = 0$.

We claim that there exists a small enough $\varepsilon > 0$ so that $x + \varepsilon y \in P$ and $x - \varepsilon y \in P$, and that would imply that x is not a vertex solution, thereby completing the proof.

To see the claim, since $Ay = 0$, we have $A(x+\varepsilon y) = Ax = b$ and similarly $A(x-\varepsilon y) = Ax = b$, satisfying equalities.

Also, since $\text{supp}(y) \subseteq \text{supp}(x)$, by choosing $\varepsilon > 0$ to be small enough, we have $x + \varepsilon y \geq 0$ and $x - \varepsilon y \geq 0$, satisfying the inequalities. So, both $x+\varepsilon y \in P$ and $x - \varepsilon y \in P$, and we are done. □

Optimal corner points

Now, we would like to prove that there is always some optimal corner point solutions.

Before we do that, we state the corresponding definitions of corner points for the inequality form, as this is usually the form that we work with.

Recall that the inequality form is $\min_x \langle c, x \rangle$ s.t. $Ax \leq b$. Let $P = \{x \mid Ax \leq b\}$ where $A \in \mathbb{R}^{m \times n}$.

The corresponding definitions are:

① Vertex solutions: x is a vertex solution if $\nexists y \neq 0$ such that $x+y \in P$ and $x-y \in P$.

② Extreme point solutions: x is the unique optimal solution for some objective function $c \in \mathbb{R}^n$.

③ Basic solutions: Let $x \in P$. We say a constraint A_i (the i-th row of A) is tight if $\langle A_i, x \rangle = b_i$.

Given $x \in P$, let $A^=$ be the submatrix of A formed by the tight constraints.

We say x is a basic solution if $A^=$ is of full rank, i.e. $\text{rank}(A^=) = n$.

We usually use ① and ③ in proofs. For completeness, we prove that they are equivalent.

Proposition: ① and ③ are equivalent.

Proposition ① and ③ are equivalent.

Proof We will prove $\neg③ \Rightarrow \neg① \Rightarrow \neg③$. Given $x \in P$, let $A^=$ be the set of rows that are tight wrt x.

$\neg③ \Rightarrow \neg①$: Suppose x is not a basic solution. By definition, the row rank of $A^=$ is less than n.

This implies that the column rank is also less than n, i.e. the columns are linearly independent.

So, $\exists y \neq 0$ such that $A^= y = 0$. Hence, $A^=(x+y) = b = A^=(x-y)$, i.e. tight constraints remain tight.

Therefore, by choosing $\varepsilon > 0$ to be small enough, the non-tight constraints (strict inequalities) won't be violated,

and thus we still have $A(x+\varepsilon y) \leq b$ and $A(x-\varepsilon y) \leq b$. hence x is not a vertex solution.

$\neg① \Rightarrow \neg③$: Suppose x is not a vertex solution. By definition, $\exists y \neq 0$ such that $A(x+y) \leq b$ and $A(x-y) \leq b$.

Let $A^=$ be the tight rows of x. i.e. $A^= x = b^=$, where $b^=$ is the corresponding right hand side.

Since $A(x+y) \leq b$ and $A(x-y) \leq b$, we must have $A^=(x+y) \leq b^=$ and $A^=(x-y) \leq b^=$, and this implies that

$A^= y \leq 0$ and $A^=(-y) \leq 0$, and hence $A^= y = 0$.

This implies that the columns of $A^=$ are linearly dependent. and so $\text{rank}(A^=) < n$, thus not basic. □

Exercise Prove that the three definitions are equivalent.

We are ready to show that there is always an optimal corner point solution.

We say the polytope P is bounded if there exists a positive integer M such that if $x \in P$,

then $|x_i| \leq M$ for all $1 \leq i \leq n$. Note that it holds for most combinatorial optimization problems as $0 \leq x_i \leq 1$.

Proposition For bounded P, for any $c \in \mathbb{R}^n$, $\exists$ basic feasible solution x such that $cx \leq cy$ for all $y \in P$.

Proof The idea is simple. We move in the tight space until we hit a corner.

If x is not basic, then $\text{rank}(A^=) < n$, and $\exists y \neq 0$ such that $A^= y = 0$.

Consider $x+\varepsilon y$ and $x-\varepsilon y$. Both solutions have the current tight constraints still tight as $A^= y = 0$.

Also, either $\langle c, x+\varepsilon y \rangle \leq \langle c, x \rangle$ or $\langle c, x-\varepsilon y \rangle \leq \langle c, x \rangle$ (or both if $\langle c, y \rangle = 0$), say $\langle c, x+\varepsilon y \rangle \leq \langle c, x \rangle$.

Set ε to be the maximum value so that $A(x+\varepsilon y) \leq b$.

Since $y \neq 0$ and P is bounded, we know that $\varepsilon < \infty$, in which case we hit a new tight constraint.

So, we can find a solution $x' = x + \varepsilon y$ with one more tight constraint if x is not basic.

This process cannot repeat forever (since we have a finite number of constraints). and hence eventually

we will reach a basic solution x^* with $\langle c, x^* \rangle \leq \langle c, x \rangle$ for any x.

In particular, if x is optimal. then x^* is an optimal solution which is also basic. □

Perfect bipartite matching

Let's run the proof on a nice example to find an integral optimal solution.

Consider the perfect bipartite matching LP: $\max_{e \in E} w_e \cdot x_e$

$$\sum_{e \in \delta(v)} x_e = 1 \quad \text{for all } v \in V$$

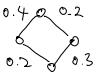
$$0 \le x_e \le 1 \quad \text{for all } e \in E.$$

We claim that a vertex solution must be integral.

Consider a fractional solution x with some edge uv having $0 < x_{uv} < 1$.

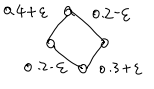
By the constraints $\sum_{e \in \delta(v)} x_e = 1$, we can find another edge vw such that $0 < x_{vw} < 1$, and so on.

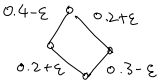
Repeat this argument until we have found a "strictly" fractional cycle.



This is an even cycle as the graph is bipartite.

Now, we can construct two fractional solutions $x+y \in P$ and $x-y \in P$ by setting





and keep the remaining variables unchanged, showing that x is not vertex.

Therefore, a vertex solution must be integral.

Next time, we will use the rank argument to prove this result again, and then to generalize to other problems.

Note that the same LP is not integral for perfect matching in general graphs.

Algorithms for solving linear programs

There are three main types of algorithms: simplex methods, ellipsoid methods, and interior point methods.

We just very briefly mention these algorithms, but highlight a feature of the ellipsoid method.

Simplex methods: Work in the equational form.

Start from an arbitrary basic feasible solution.

Move to a "neighboring" basic solution by removing one column and adding one column.

If there is a neighbor with objective value as good, choose one and go there.

If all neighbors have worse objective values, stop and return the current solution, and this is correct because a local optimal solution in a convex optimization problem is a global optimal solution.

There are many different rules in choosing which neighbor to go, and some may run into infinite loops.

For most natural rules, there are examples showing that they need exponentially many steps to reach an optimal solution, even for some randomized rules.

It is still a major open problem whether there are polynomial time simplex algorithms.

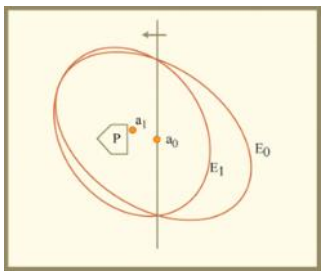
Actually, one necessary condition is whether every polytope has diameter at most polynomial in n and m, and this is known as Hirsch's conjecture and it is still wide open.

One important remark is that many combinatorial algorithms (e.g. augmenting path algorithms) can be understood as a simplex algorithm of a specific rule.

In practice, Simplex algorithms seem to be very competitive, and there is a theory of Smoothed Analysis developed to explain this phenomenon.

Ellipsoid algorithm: It is the first known polynomial time algorithm for LP, discovered by Khachiyan.

First, note that the optimization problem can be reduced to a decision problem of checking whether P is non-empty.



Initially, we find a large enough ellipsoid to guarantee that it contains the whole polytope P.

In each step, we check if the center a_i of the current ellipsoid is in P.

If yes, we are done.

If not, the algorithm requires a hyperplane that seperates a_i from P.

Given such a hyperplane H, the algorithm would find a smallest ellipsoid E_{i+1} that contains $E_i \cap H$, in particular it still guarantees that $P \subseteq E_{i+1}$.

Repeat until $\text{vol}(E_i)$ has become too small, not possible to contain P.

The key of the analysis is to show that the volume of the ellipsoids decreases fast enough.

An elementary analysis shows that $\text{vol}(E_{i+1}) / \text{vol}(E_i) \leq e^{-1/2(n+1)}$, so that in $O(n)$ iterations the volume of the ellipsoid is decreased by a constant factor.

This method falls in the class of algorithms called cutting plane methods, not necessarily using ellipsoids.

Interior point methods This is another class of polynomial time algorithms for solving LP, discovered by Karmarkar.

Roughly, the linear program is reduced to an unconstrained optimization problem, by using "barrier functions" to ensure that an optimal solution of this unconstrained problem stays in the polytope.

Initially, we start from the "center" of the polytope.

In each step, we slowly decreaase the "force" of the barrier functions, resolve an unconstrained optimization problem using Newton method, and move closer to an optimal solution of the original problem.

This is competitive with Simplex methods in practice.

In theory, there are recent exciting progress giving fast interior point algorithms for combinatorial optimization problems, including matchings and maximum flows.

Optimization via Separation

An important feature of the ellipsoid method is that it only requires a "separation oracle" for it to work. That is, an algorithm to decide whether $x \in P$; if not, return a separating hyperplane.

In particular, the ellipsoid method does not require us to write down the LP explicitly, and in principle we could solve an exponential sized LP as long as there is a polynomial time separation oracle.

Let's see a nontrivial example of solving an exponential sized LP.

Consider the spanning tree polytope.

$$\min \sum_{e \in E} c_e \cdot x_e$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \forall\, S \subsetneq V$$, where $E(S)$ denotes the sets of edges with both endpoints in S.

$$\sum_{e \in E} x_e = |V| - 1$$

$$x_e \geq 0 \qquad \forall\, e \in E$$

It is not difficult to see that it is a relaxation of the minimum spanning tree problem.

For any integral solution, the first class of constraints say that any subset S cannot contain more than $|S|-1$ edges, and this is used to forbid any cycles.

The second constraint says we need to choose exactly $n-1$ edges, and we know that an acyclic subgraph with $|V|-1$ edges must be a spanning tree.

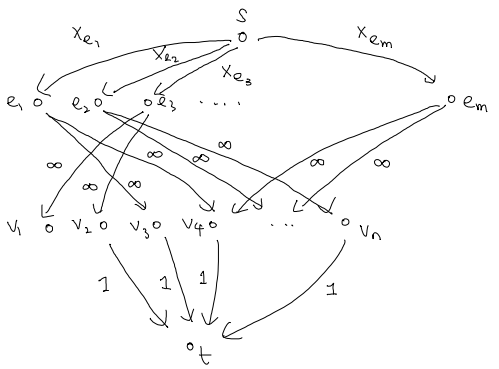
We will see later that this LP is indeed integral.

To solve this exponential sized LP, we won't write it down. Instead, we need to design a polynomial time algorithm to determine if $x \in P$, and if $x \notin P$ we must return a constraint that x does not satisfy.

The constraint $\sum_{e \in E} x_e = |V| - 1$ is easy to check. So we focus on the class $\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall\, S \subsetneq V$.

Given an LP solution $x \in \mathbb{R}^{|E|}$, we will run $|V|$ min-cut s-t algorithms, each to check whether there is a violating set containing a specific vertex v_i.

The construction is shown in the following picture.



one vertex for each edge e_i,
the capacity of the arc se_i is $x(e_i)$.

one vertex for each vertex v_i,
say $e_3 = (v_1, v_2)$, then we add two arcs (e_3, v_1) and (e_3, v_2), each of capacity ∞

except for the special vertex, in this case v_1, there is an edge from v_i to t with capacity one.

Lemma There is a violating set containing v_1 iff the min s-t cut value is strictly less than $|V|-1$.

Proof If there is a violating set containing v_1 say $S = \{v_1, v_2, \ldots, v_s\}$ such that $\sum_{e \in E(S)} x_e > |S| - 1$.

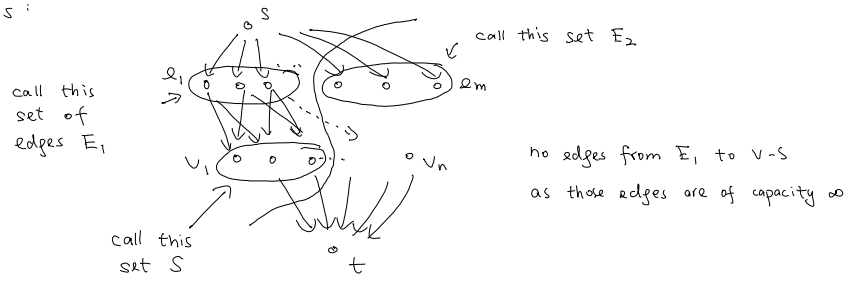
Note that each edge $e \in E(S)$ must send its flow to vertices in S, but $\sum_{e \in E(S)} x_e > |S| - 1$, and S has only $|S|-1$ capacity to t, and so the capacity in $E(S)$ cannot be sent to the sink t, and

thus the s-t flow value would be strictly smaller than $|V|-1$, so min s-t cut $< |V|-1$.

For the other direction, consider a s-t cut of value strictly smaller than $|V|-1$.

Consider such a cut. It cannot cut any edge of capacity ∞. Also, the cut cannot contain all top edges (since $\sum_{e \in E} x_e = |V|-1$) and cannot contain all bottom edges (since there are $|V|-1$ edges).

So, the cut should look like this:



So, all edges in E_1 must have both vertices in S, and hence $\sum_{e \in E(S)} x_e \geq \sum_{e \in E_1} x_e$.

We claim that $\sum_{e \in E_1} x_e > |S| - 1$, and this would imply that S is a violating set containing v_1.

To see this, note that $|V|-1 >$ s-t cut value $= |S|-1 + \sum_{e \in E_2} x_e = |S|-1 + |V|-1 - \sum_{e \in E_1} x_e$.

Therefore, $\sum_{e \in E(S)} x_e \geq \sum_{e \in E_1} x_e > |S|-1$, and this completes the proof. $\square$

So, to do this construction for every special vertex v_i, if all min s-t cut values are $|V|-1$, then x is feasible; otherwise if some s-t cut value is $|V|-1$, we find a violating constraint.

To conclude, we can use ellipsoid algorithm to solve this exponential sized LP in polynomial time.

References

- Iterative methods in combinatorial optimization.
- Understanding and using linear programming, by Matousek (highly recommended author).
- Combinatorial optimization: polyhedra and efficiency, by Schrijver
  (3-volume set master piece, >1800 pages of concise proofs!)