CS602 Applied Algorithms **2019-20 Sem II**

Lecture 1: January 13

Scribe: Harmeen Kaur *Lecturer: Rohit Gurjar*

1 Contents of the Course

1. Linear Programming Basics:
   - Duality
   - Primal Dual Approach
2. Applications in Combinatorial optimization:
   - Matching
   - Flow
   - Cuts
   - Matroids
3. Submodular optimization
4. Approximation Algorithms:
   - Vertex cover
   - Disjoint paths
   - Scheduling
5. Online Algorithms:
   - Matching
   - Load Balanicing
   - Semi Definite Programming (SDP)
   - Continuous methods

2 Linear Programming

Linear Programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints.

It can be described as follows:

- we are given n variables: $x_1, x_2, x_3 .... x_n \in \mathbb{R}$.
- m linear inequalities in these variables (equalities are OK too)
- And we are supposed to optimise some linear function

  $$f(x_1, x_2, x_3 .... x_n) = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

  in these variables, subject to the given linear constraints.

Example 1: Let there be a set of variables in 2 dimensional space

$$(x_1, x_2) \in \mathbb{R}^2$$

which are subjected to the following constraints:

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_1 + x_2 \leq 1$$

The goal is to maximize a linear objective function $f(x_1, x_2) = 2x_1 + x_2$. One can verify that $(x_1 = 1, x_2 = 0)$ is the point that satisfies the above constraints and achieves the maximum function value, $f(1, 0) = 2$.

Example 2 (Transportation Problem): Let there be k coal mines $M_1, M_2, \ldots, M_k$ and their respective capacities of production be $c_1, c_2, c_3, \ldots, c_k$. Let there be ℓ industries $P_1, P_2, P_3, \ldots, P_\ell$ having demands of coal from mines $d_1, d_2, d_3, \ldots, d_\ell$ respectively.

$p_{i,j}$: Transportation cost per unit from M_i to P_j (given as part of the problem).

The Objective is to minimize overall cost of transportation fulfilling all the demands of industries. Transportation problem can be expressed as a problem of Linear Programming. The linear program will use $k\ell$ variables $\{x_{i,j} \mid 1 \leq i \leq k,\ 1 \leq j \leq \ell\}$, where

$x_{i,j}$: amount of coal transferred from M_i to P_j

- for each $1 \leq i \leq k$

$$\sum_{j=1}^{l} x_{i,j} \leq c_i$$

- for each $1 \leq j \leq l$

$$\sum_{i=1}^{k} x_{i,j} = d_j$$

- The goal is to minimize:

$$\sum_{i,j} p_{i,j} x_{i,j}.$$

The above problem deals with continuous variables as the amount of coal transferred can be an arbitrary real number. Unlike this situation, in most combinatorial optimization problems our domain is actually discrete. Still linear programming proves to be a useful tool in combinatorial optimization. Let us try to express such a discrete problem via a linear program.

Example 3 (Largest Independent Set): Given a graph, the problem asks to find the largest set of vertices that have no common edge between them. Let us try to express this problem as an optimization problem with linear constraints. Let $G = (V, E)$ be the given graph where, V is the vertex set and $|V| = n$ and E is the edge set. Consider a variable x_i for each vertex v_i such that

$$x_1, x_2, \ldots, x_n \in \{0, 1\}.$$

Here, $x_i = 1$ is supposed to mean that we select vertex v_i in the set and $x_i = 1$ means we do not select it. Now, let us put the "no common edge between two vertices in the set" condition as a linear inequality.

$$\forall e = (v_i, v_j) \in E,\ x_i + x_j \leq 1.$$

The above constraint is simply expressing that for any edge, we are allowed to take at most one of its end points in the set. The objective is to maximise:

$$\sum_{i=1}^{n} x_i.$$

One can verify that the maximizing solution will give us the largest independent set.

Since the variables here are bound to take values from a discrete set of $\{0, 1\}$, this is not a linear program but an integer program.

Linear Programming(LP) Relaxation: The relaxation of a (mixed) integer linear program is the problem that arises by removing the integrality constraint of each variable. For example, in a 0-1 integer program, all constraints are of the form

$$x_i \in \{0, 1\}$$

The relaxation of the original integer program instead uses a collection of linear constraints:

$$0 \leq x_i \leq 1$$

The resulting relaxation is a linear program, hence the name. This relaxation technique transforms an NP-hard optimization problem (integer programming) into a related problem that is solvable in polynomial time (linear programming).[1]

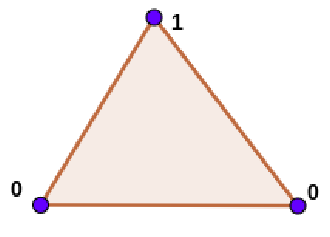
Example: So in the above example of largest independent set if we apply relaxation technique to transform it to a linear programming problem, it will look like: Consider a variable x_i for each vertex v_i such that :

$$\forall i,\ 0 \leq x_i \leq 1$$
$$\forall e = (v_i, v_j) \in E, x_i + x_j \leq 1$$

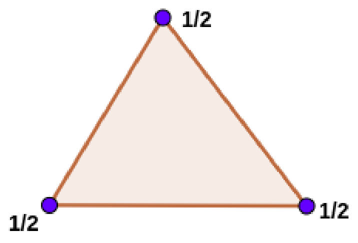
- The objective is to maximise:

$$\sum_{i=1}^{n} x_i$$

As the definition states, NP-hard problems after relaxation transforms to a related problem. Since linear programming problems can be solved in polynomial time, that related problem is not exactly equivalent to the original NP-hard problem. For above example of the largest independent set problem, consider the complete graph on three nodes. The following figure shows that the optimal values of the integer program and the corresponding linear program are different.



(a) Integer Programming Problem ($OPT = 1$)



(b) Linear Programming Problem ($OPT = 3/2$)

Here the largest independent set is of size 1 but, the corresponding linear programming relaxation has maximum value 3/2.

3 Various forms of linear programs

General form of a linear program: For given $\{a_{i,j} \in \mathbb{R} \mid 1 \leq i \leq m + m',\ 1 \leq j \leq n\}$, $\{b_i \in \mathbb{R} \mid 1 \leq i \leq m + m'\}$, and $\{c_j \in \mathbb{R} \mid 1 \leq j \leq n\}$, we want to find $x_1, x_2, \ldots, x_n \in \mathbb{R}$ such that

$$\text{for } 1 \leq i \leq m, \quad \sum_{j=1}^{n} a_{i,j} x_j \geq b_i,$$

$$\text{and for } m+1 \leq i \leq m', \quad \sum_{j=1}^{n} a_{i,j} x_j = b_i,$$

and which maximizes

$$\sum_{j=1}^{n} c_j x_j.$$

Note that '$\leq$' kind of inequality can be converted to '$\geq$' kind of inequality (and vice versa) by multiplying with -1. Also, maximization can be converted to minimization (and vice versa) by multiplying with -1.

Standard Forms: There are various standard forms to represent linear programs, which are as expressive as the general form. We present two of these and argue that they can be converted to one another.

1. minimize/maximize
$$\sum_{j=1}^{n} c_j x_j$$
such that for $1 \leq i \leq m$
$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i$$
This actually gives intersections of half spaces formed by inequalities.

2. minimize/maximize
$$\sum_{j=1}^{n} c_j x_j$$
such that
   - for $1 \leq j \leq n$,
$$x_j \geq 0$$
   - and for $1 \leq i \leq m$
$$\sum_{j=1}^{n} a_{ij} x_j = b_i$$

Converting one form to another:

- Form 1 to 2 (shown by examples):

  Form 1:
$$x_1 + x_2 \geq 3$$
  Equivalent Form 2 :
$$x_1 + x_2 = y_1 + 3,\ y_1 \geq 0$$
  (converted inequality to equivalent equality by introducing a new variable).

  Note that in form 2, all variables have non-negativity constraints, while this need not be true form 1. We replace any such variable x_i with $z_i - u_i$, where z_i, u_i are two new variables and put non-negativity constraints
$$z_i \geq 0,\ u_i \geq 0$$

- Form 2 to 1: Taking any general form 2 set of liner program, keeping the min/max objective the same and all $x_j \geq 0$ conditions the same, replace all equality constraint by two inequalities as:

  $$\sum_{j=1}^{n} a_{ij}x_j = b_i$$

  can be replaced by:

  $$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

  ,

  $$-(\sum_{j=1}^{n} a_{ij}x_j) \geq -b_i$$

4 Geometric Perspective

Definition 1.1 (Polyhedron) *A polyhedron is the feasible region of a given set of linear inequalities. That is, for given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, the following set is called a polyhedron*

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

Example: In space $\mathbb{R}^2$, say the constrains are as follows:

$$2x_1 + x_2 \leq 2$$
$$x_1 + x_2 \geq 0$$
$$x_1 \geq 0$$

The yellow region in Figure 1 shows the corresponding polyhedron.

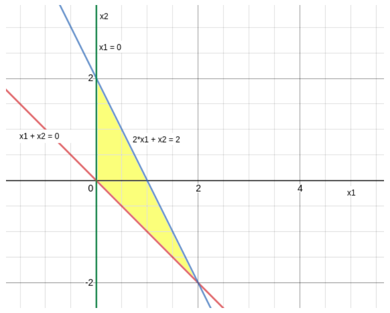


Figure 1: A Polyhedron

We will often deal with polyhedron which are bounded.

Definition 1.2 (Polytope) *A set $S \subseteq \mathbb{R}^n$ is called bounded if $\exists r > 0$ such that whole set S is inside the the hypersphere of radius r centered at origin. In other words, every point p in the set S satisfies $\|p\|_2 \leq r$. A bounded polyhedron is called a polytope.*

Example: A cube is a polytope. Figure 2 shows the cube defined by the following constraints.

$$0 \leq x_1 \leq 1$$
$$0 \leq x_2 \leq 1$$
$$0 \leq x_3 \leq 1$$

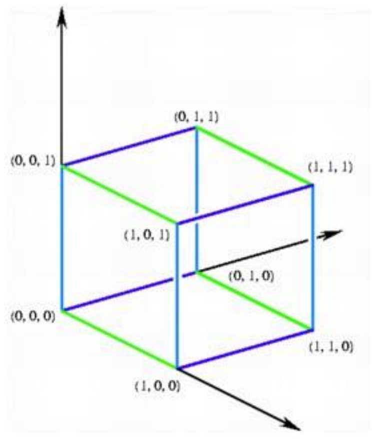


Figure 2: Cube [2]

Facets: In an n dimensional polyhedron, its facets are $n-1$ dimensional hyperplane (segments) bounding the polyhedron. For example, in case of a 3 dimensional cube, its 2-D faces are facets.

Definition 1.3 (Face) *Any dimensional boundary of a polyhedron is called a face. e.g. vertex, edge , or the polyhedron itself. We can get a face by replacing any subset of inequalities with equivalent equality.*

Example:

$$2x_1 + x_2 \leq 2 \tag{1}$$
$$x_1 + x_2 \geq 0 \tag{2}$$
$$x_1 \geq 0 \tag{3}$$

Now replace inequality (3) with its corresponding equality. The new set of constraints will look like:

$$2x_1 + x_2 \leq 2 \tag{4}$$
$$x_1 + x_2 \geq 0 \tag{5}$$
$$x_1 = 0 \tag{6}$$

which forms line/edge of the polyhedron $x_1 = 0$, which is a face.

References

[1] source :https://en.wikipedia.org/wiki/Linear_programming_relaxation

[2] source : www.math.brown.edu/ banchoff/Beyond3d/chapter8/section01.html