COL758: Advanced Algorithms **Spring 2019**

Lecture 2: January 7

Lecturer: Naveen Garg *Scribe: Praneeth Kacham*

2.1 Linear Programming

In last class, we have seen what a linear program is and a few techniques to solve a linear program. Following is the canonical form of a linear program we are going to use throughout the course.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$.
Following are few techniques to solve linear programs.

1. Simplex Method - Exponential time
2. Ellipsoid Algorithm - Polynomial time
3. Interior Point Algorithms - Polynomial time

2.2 Flows

Notation

$$\delta_{in}(v) = \{e = (x, v) | e \in E\} : \text{all arcs coming into vertex } v$$

$$\delta_{out}(v) = \{e = (v, x) | e \in E\} : \text{all arcs going out of vertex } v$$

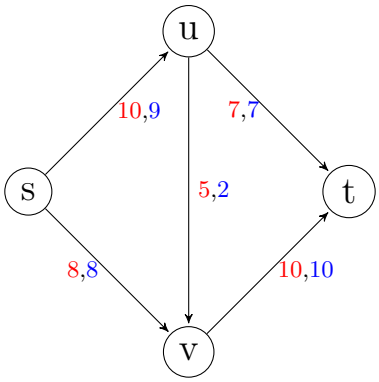
Definition of *flow*

Given a directed graph $G = (V, E)$, a capacity function $c : E \to \mathbb{R}^+$, source vertex $s \in V$ and destination vertex $t \in V$, a function $f : E \to \mathbb{R}^+$ is called a *flow* if it satisfies the following conditions

1. Capacity Constraint : $\forall e \in E : \ f(e) \leq c(e)$
2. Flow Conservation : $\forall v \neq s, t \in V : \ \sum_{e \in \delta_{in}(v)} f(e) = \sum_{e \in \delta_{out}(v)} f(e)$

Example of a valid flow

Red denotes the capacity of an edge and Blue denotes the flow on an edge.



Flow Maximization

Given a graph $G = (V, E)$, capacities c and vertices s, t, a flow f is maximum flow if the net flow from s to t is maximum among all the flows. Net flow from s to t in a flow f is given by

$$\sum_{e \in \delta_{out}(s)} f(e) - \sum_{e \in \delta_{in}(s)} f(e)$$

Definition of $s - t$ *cut*

Partition of the vertex set V is called a *cut*. A partition of vertex set V into (X, Y) such that $s \in X$ and $t \in Y$ is called an $s - t$ cut.

Capacity of a *cut*

Capacity of an $s - t$ cut is defined as follows

$$\text{capacity of } s-t \text{ cut } (X,Y) = \sum_{\substack{e=(u,v): \\ u \in X, v \in Y}} c(e)$$

It is easy to observe that

$$\text{Max flow} \leq \text{Min cut}$$

We will prove later that

$$\text{Max flow} = \text{Min cut}$$

Max flow problem as a linear program

We shall see how to model max-flow problem as a linear program. For every edge we have a variable x_e which denotes the flow on the edge e. Capacity constraints enforce the following constraint on x_e

$$\forall e \in E \quad x_e \leq c_e$$

Flow conservation condition introduces the following constraints

$$\forall v \neq s, t \in V \quad \sum_{e \in \delta_{in}(v)} x_e = \sum_{e \in \delta_{out}(v)} x_e$$

Non-negativity of the flow implies

$$\forall e \in E \quad x_e \geq 0$$

Net flow from s to t is given by

$$\sum_{e \in \delta_{out}(s)} x_e - \sum_{e \in \delta_{in}(s)} x_e$$

Thus, maximum flow problem can be encoded by the following linear program

$$\begin{aligned}
\max \quad & \sum_{e \in \delta_{out}(s)} x_e - \sum_{e \in \delta_{in}(s)} x_e & & \\
\text{s.t.} \quad & x_e \leq c_e & & \forall e \in E \\
& \sum_{e \in \delta_{in}(v)} x_e = \sum_{e \in \delta_{out}(v)} x_e & & \forall v \neq s, t \in V \\
& x_e \geq 0 & & \forall e \in E
\end{aligned}$$

Note

It can be seen that max-flow of a graph removing all the arcs coming into the vertex s is equal to max-flow of the original graph.

Path decomposition of flow

Another way to look at a flow is considering paths from s to t and sending some amount of flow through each path. This automatically takes care of flow-conservation constraint. Let $\mathcal{P}$ be the set of all s-t paths. Capacity constraint can be given as follows:

$$\forall e \in E \sum_{P \in \mathcal{P}: e \in P} f(P) \leq c_e$$

Decomposing a given flow f into flow along paths

Result: *paths* : Flow f decomposed into paths
$paths \leftarrow \phi$;
while *There is a path from s to t without zero-flow edges according to f* **do**
  $p \leftarrow$ any path from s-t without zero-flow edges according to f;
  $f_p = \min_{e \in p} f_e$;
  $\forall e \in p \quad f_e \leftarrow f_e - f_p$;
  $paths \leftarrow paths \cup (p, f_p)$;
end

Algorithm 1: Path decomposition of flow

For any valid flow, the above algorithm terminates in at most $|E|$ steps and at the end of the algorithm, f_e is zero for each edge. The latter statement can be proved using the flow-conservation constraint. Thus, any valid flow can be decomposed into flow along at most $|E|$ paths.

Max flow LP in terms of paths

$$\begin{aligned}
\text{max.} & \sum_{P \in \mathcal{P}} f_P \\
& \sum_{P \in \mathcal{P}: e \in P} f_P \leq c_e \quad \forall e \in E \\
& f_P \geq 0
\end{aligned}$$

Note that this formulation may have exponential number of variables but this formulation is useful when proving Max-Cut=Min-Flow and multi-commodity flow problem.

2.3 Dual of a linear program

Motivation

Consider the following linear program

$$\begin{aligned} \max.\ & 4x_1 + x_2 + 5x_3 + 3x_4 \\ & x_1 - x_2 - x_3 + 3x_4 \leq 1 \\ & 5x_1 + x_2 + 3x_3 + 8x_4 \leq 55 \\ & -x_1 + 2x_2 + 3x_3 - 5x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Consider the following method of finding an upper bound on the optimum. Any feasible solution must satisfy all the constraints. Adding the 2nd and 3rd constraints, we obtain $4x_1 + 3x_2 + 6x_3 + 3x_4 \leq 58$. Each of the coefficients of x_i in this constraint is greater than the coefficient of x_i in the objective. From non-negativity of $x_i's$, we obtain

$$4x_1 + x_2 + 5x_3 + 3x_4 \leq 4x_1 + 3x_2 + 6x_3 + 3x_4 \leq 58$$

The above holds for any feasible solution. Hence, 58 is an upper bound on the optimum.

How do we obtain the best upper-bound using the linear combinations of constraints? Let ith constraint be multiplied by $\lambda_i \geq 0$ and add all of them. We obtain

$$\lambda_1(x_1 - x_2 - x_3 + 3x_4) + \lambda_2(5x_1 + x_2 + 3x_3 + 8x_4) + \lambda_3(-x_1 + 2x_2 + 3x_3 - 5x_4) \leq \lambda_1 + 55\lambda_2 + 3\lambda_3$$

Collecting coeffecients of each x_i,

$$(\lambda_1 + 5\lambda_2 - \lambda_3)x_1 + (-\lambda_1 + \lambda_2 + 2\lambda_3)x_2 + (-\lambda_1 + 3\lambda_2 + 3\lambda_3)x_3 + (3\lambda_1 + 8\lambda_2 - 5\lambda_3)x_4 \leq \lambda_1 + 55\lambda_2 + 3\lambda_3$$

A sufficient condition for $\lambda_1 + 55\lambda_2 + 3\lambda_3$ to be an upperbound to optimum is that coefficient of each of the $x_i's$ is greater than the corresponding coefficient in the objective. Obtaining the best upperbound in this method can be encoded as a minimization problem as follows

$$\begin{aligned} \min.\ & \lambda_1 + 55\lambda_2 + 3\lambda_3 \\ & \lambda_1 + 5\lambda_2 - \lambda_3 \geq 4 \\ & -\lambda_1 + \lambda_2 + 2\lambda_3 \geq 1 \\ & -\lambda_1 + 3\lambda_2 + 3\lambda_3 \geq 5 \\ & 3\lambda_1 + 8\lambda_2 - 5\lambda_3 \geq 3 \\ & \lambda_1, \lambda_2, \lambda_3 \geq 0 \end{aligned}$$

Any feasible solution for this linear program gives an upper bound on the optimum and optimum of this linear program is the best upper bound that can be obtained based on using the linear combinations of constraints. This optimization problem is called *dual* problem and the original optimization problem is called *primal* problem.

Dual problem of the canonical LP

Using the previous ideas, let us write out the dual program of the linear optimization problem in canonical form.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Let the vector of multipliers be given by $\lambda \in \mathbb{R}^m$(One for each constraint). Dual program of this is as follows.

$$\begin{aligned} \min \quad & b^T \lambda \\ \text{s.t.} \quad & A^T \lambda \geq c \\ & \lambda \geq 0 \end{aligned}$$

Weak duality Theorem

Let x be any feasible solution for the primal and λ be any feasible solution for dual then

$$c^T x \leq b^T y$$

and hence

$$OPT(Primal) \leq OPT(Dual)$$

This can be seen easily from the way we have defined dual of a linear program.

Strong duality Theorem

Strong duality theorem states that

$$OPT(Primal) = OPT(Dual)$$

and there exists primal optimal solution x^* and dual optimal solution λ^* such that

$$c^T x^* = b^T \lambda^*$$

Complementary Slackness

Consider the following terms for primal feasible x and dual feasible λ,

$$c^T x \quad \lambda^T A x \quad \lambda^T b$$

For any feasible x for the primal problem, we have

$$Ax \leq b$$

Now, λ being a non-negative vector, we have

$$\lambda^T A x \leq \lambda^T b$$

For any feasible λ for dual problem, we have

$$A^T \lambda \geq c$$

Now, x being a non-negative vector(Since, primal feasible), we have

$$x^T A^T \lambda \geq x^T c$$

same as

$$\lambda^T A x \geq c^T x$$

Thus, for any x which is primal feasible and λ which is dual feasible, we have

$$c^T x \leq \lambda^T A x \leq b^T \lambda$$

This a proof of weak-duality theorem. The above is also valid for x^* and λ^*. Hence,

$$c^T x^* \leq \lambda^{*T} A x^* \leq b^T \lambda^*$$

But, strong duality states that $c^T x^* = b^T \lambda^*$. Hence,

$$c^T x^* = \lambda^{*T} A x^* = b^T \lambda^*$$

Consider the equality $c^T x^* = \lambda^{*T} A x^*$. This implies, $(c^T - \lambda^{*T} A)x^* = 0$. Now, from dual feasibility, $c - A^T \lambda^*$ is a non-negative vector and from primal feasibility, x^* is a non-negative vector. Dot product being 0 implies that for each i, either ith component of $(c - A^T \lambda^*)$ is 0 or ith component of x^* is 0 i.e., either $A_i^T \lambda^* = c_i$ or $x_i^* = 0$. Similarly, either $\lambda_i^* = 0$ or $A_i^T x^* = b_i$.

We can group the constraints of primal problem and dual problem as follows.

$x_i \geq 0$ & constraint corresponding to x_i in dual

$\lambda_i \geq 0$ & constraint in primal problem of which λ_i is the multiplier

For optimal primal and dual solutions atmost one of the inequalities in each of the pairs as defined above can have slack (i.e., not tight). Hence, this property is called *complementary* slackness.