COL758: Advanced Algorithms	**Spring 2019**

Lecture 3: January 10

Lecturer: Naveen Garg — *Scribe: Jai Moondra*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*
Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

3.1 Previous lecture

Linear program duality:

1. How to write a dual
2. Strong and weak duality theorems
3. Complementary slackness

Max-flows:

1. Path decomposition
2. LP for max-flow using paths

3.2 Path LP for max-flow and its dual

The max-flow problem

Given a directed graph $G = (V, E)$, $s, t \in V$, and positive edge capacities $c : E \to \mathbb{R}^+$, we define a *flow* in the graph to be a function $f : E \to \mathbb{R}^+ \cup \{0\}$ such that

- $f(e) \leq c_e \quad \forall e \in E$ (capacity constraint)
- $\sum_{e \in \delta_{in}(v)} f(e) = \sum_{e \in \delta_{out}(v)} f(e) \quad \forall v \in V - \{s, t\}$ (flow conservation)

Given a flow function f, the *total flow* from s to t was defined as

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

The problem is to maximize the total flow.

We saw in the previous lecture that every flow can be decomposed into flow along $s - t$ paths. Denote by $\mathcal{P}$ the set of all $s - t$ paths.

The linear program and its dual

We saw the following linear program for the max-flow problem in the previous lecture: for every path $p_i \in \mathcal{P}$, we have a variable f_i that denotes the flow along path p_i:

$$\max \sum_{p_i \in \mathcal{P}} f_i$$

$$\text{subject to} \sum_{i : e \in p_i} f_i \leq c_e \quad \forall e \in E$$

$$f_i \geq 0 \quad \forall i$$

For the dual of this linear program, we introduce a variable l_e for every edge e, corresponding to the appropriate constraint. Then, the dual is:

$$\min \sum_{e \in E} c_e l_e$$

$$\text{subject to} \sum_{e \in p_i} l_e \geq 1 \quad \forall p_i \in \mathcal{P}$$

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

We can interpret this linear program as assigning a non-negative length to each edge such that the length of the shortest $s-t$ path (that is, the sum of lengths of the edges on this path) under this length function is 1. Under these constraints, we seek to minimize the value $\sum_e c_e l_e$.

Cuts and $0-1$ solutions to the dual

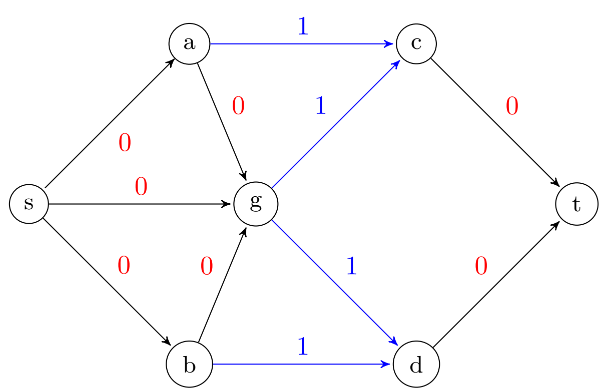
Given any $s-t$ cut $(S, V-S)$ (which we call the cut *formed* by S), assign $l_e = 1$ for each edge e one of whose end-points lies in S and the other in $V-S$ (we say that these edges *form the cut* $(S, V-S)$), assign $l_e = 1$. For all other edges e, assign $l_e = 0$.

Now, for every $s-t$ cut, every $s-t$ path must have an edge e that is a part of the cut. This means that the solution we created above to the dual is a feasible one, since by our assignment, every $s-t$ path now has length at least 1.

This means that every cut corresponds to a $0-1$ feasible solution to the dual, and the size of the cut (the total cost of edges forming the cut) is an upper bound on the optimum for the dual, which we will call OPT.

We will now show that in fact, there exists an optimum solution to the dual which corresponds to a cut. By strong duality, this will imply that the max-flow in the graph is equal to the min-cut.

In the following example, the cut edges are in blue, and are assigned $l_e = 1$. One can observe that this is a feasible solution to the dual because every $s-t$ path contains at least one of the four edges forming the cut and hence has length at least 1.



We now claim the following: given any feasible solution to the dual with objective function value α, we can find a cut with cost β such that $\beta \leq \alpha$. But β is also the objective function value for the corresponding $0-1$ solution.
Hence, we conclude that there must be an optimum dual solution that corresponds to a cut.

3.3 Optimal solution that is a cut

Convex decomposition of any feasible solution into cuts

For any cut C_i, let the cost of this cut (the sum of costs of edges across the cut) be c_i. Suppose we are given a feasible solution l to the dual. Let the cost of this solution be c. We will show that this feasible solution can be 'decomposed' into cuts $C_1, C_2, \ldots, C_t$ such that

$$c = \lambda_1 c_1 + \lambda_2 c_2 + \ldots + \lambda_t c_t$$

where each $\lambda_i \in (0, 1]$ and $\sum_{i=1}^{t} \lambda_i = 1$. This is called a convex decomposition of c into $c_1, c_2, \ldots, c_t$.

Now, there exists at least one i such that $c_i \leq c$ because otherwise,

$$c = \sum_{i=1}^{t} \lambda_i c_i > \sum_{i=1}^{t} \lambda_i c = c \sum_{i=1}^{t} \lambda_i = c$$

This implies that there is a cut C_i such that $c_i \leq c$. Therefore, the corresponding $0-1$ solution to the cut C_i is at least as good as the feasible solution.

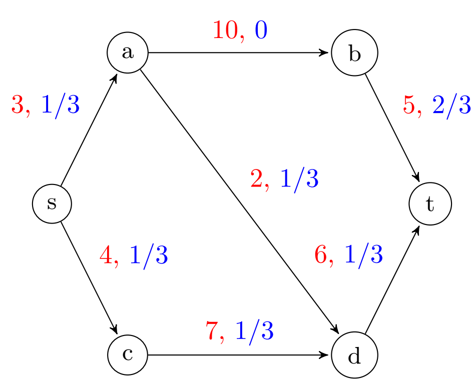
In particular, this means that there is a cut for which the corresponding $0-1$ solution is an optimum solution. Let this cut be called C^*. Our claim implies that there can be no other cut whose cost is less than cost(C^*).

Strong duality states that the optimum value for the path max-flow LP is equal to the optimum value for its dual. The optimum value for the dual is cost(C^*), and the optimum value for the primal is the value of max-flow in the graph. Thus, our claim also implies that cost(C^*) is equal to the max-flow in the graph, thus proving the max-flow min-cut theorem.

We now illustrate with example an algorithm which achieves this convex decomposition (without proof). We have a continuous notion of time and we start at s at time 0, and traverse the edges of the graph as in Dijkstra's algorithm, under the lengths l_e of the solution to the dual.

We maintain a cut at each time t, which is simply formed by the set of vertices we have reached upto point t. As we reach new vertices, we keep adding them to the cut, thus updating the cut. For any cut C_i, the corresponding coefficient λ_i is the length of the time interval for which we had that cut in the algorithm.

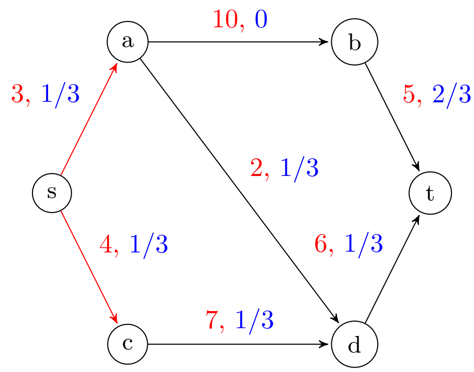
Given the solution l, first assign the length l_e to edge e for all $e \in E$. The lengths l_e's are in blue while the edge capacities are in red.



The total cost of this solution is

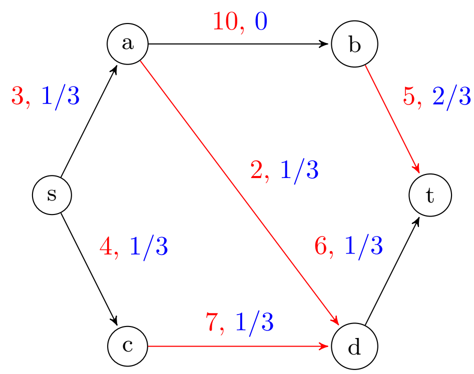
$$c = 3 \times \frac{1}{3} + 10 \times 0 + 5 \times \frac{2}{3} + 4 \times \frac{1}{3} + 7 \times \frac{1}{3} + 2 \times \frac{1}{3} + 6 \times \frac{1}{3} = \frac{32}{3}$$

Now, we begin at s at time 0. We move outward from s in all directions, covering length Δt in time Δt. Until time $t = 1/3$, we stay in the cut corresponding formed by $\{s\}$ (denoted in red):

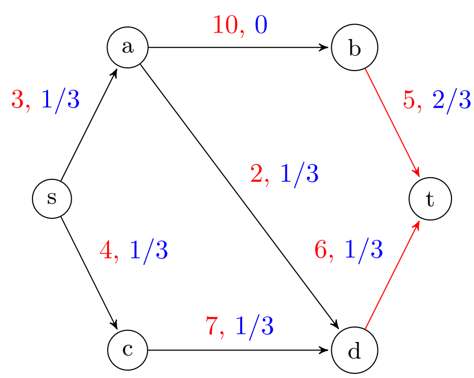


At time $t = 1/3$, we reach a, and since $l_{ab} = 0$, we also immediately reach b, and we have also reached c so that the cut at this time is $\{s, a, b, c\}$.

From time $t = 1/3$ to time $t = 2/3$, the cut is formed by $\{s, a, b, c\}$, as shown:



At time $t = 2/3$, we reach vertex d. From time $t = 2/3$ to time $t = 1$, the cut is formed by $\{s, a, b, c, d\}$:



Therefore, the solution with cost c can be decomposed into these three cuts $C_1 = \{s\}, C_2 = \{s, a, b, c\}, C_3 = \{s, a, b, c, d\}$, such that

$$c = \left(\frac{1}{3} - 0\right)c_1 + \left(\frac{2}{3} - \frac{1}{3}\right)c_2 + \left(1 - \frac{2}{3}\right)c_3 = \frac{1}{3}c_1 + \frac{1}{3}c_2 + 1\frac{1}{3}c_3$$

The weight of a cut comes from the total time for which that particular cut was being traversed during the algorithm.

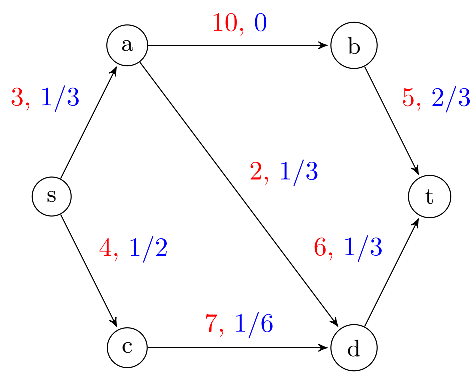
And in fact, we can conclude the above equation from something stronger. We can write every solution l as a 7-dimensional vector, where the order for the edges is as follows:

$$\begin{bmatrix} sa \\ ab \\ bt \\ sc \\ cd \\ ad \\ dt \end{bmatrix}$$

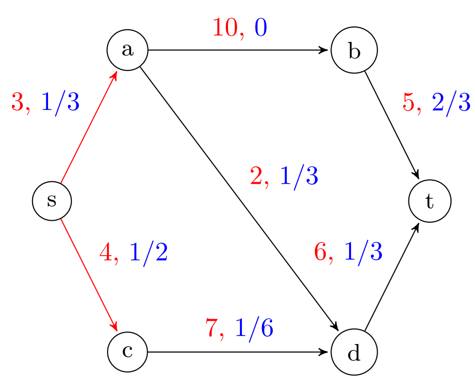
We can then write the convex decomposition in the following way:

$$\begin{bmatrix} 1/3 \\ 0 \\ 2/3 \\ 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

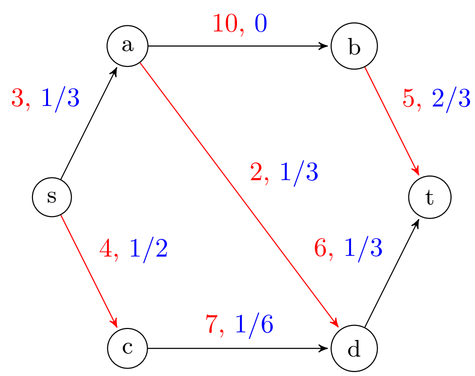
As another example, we consider another feasible solution l on t, and write the corresponding convex decomposition:



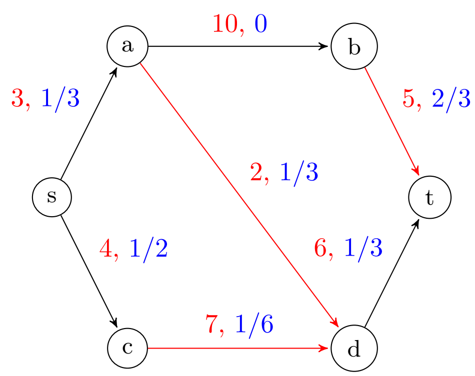
From the time $t = 0$ to $t = 1/3$, the cut is formed by $\{s\}$:



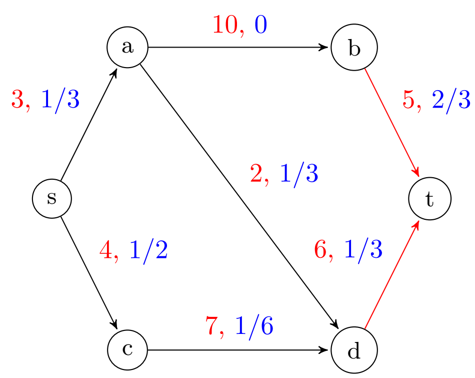
At time $t = 1/3$, node b comes in the cut since $l_{ab} = 0$. From time $t = 1/3$ to $t = 1/2$, the cut is formed by $\{s, a, b\}$:



From time $t = 1/2$ to time $t = 2/3$, the cut is formed by $\{s, a, b, c\}$:



From time $t = 2/3$ to time $t = 1$, the cut is formed by $\{s, a, b, c, d\}$:



We can write the corresponding matrix equation:

$$\begin{bmatrix} 1/3 \\ 0 \\ 2/3 \\ 1/2 \\ 1/6 \\ 1/3 \\ 1/3 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$