LINEAR OPTIMIZATION

LECTURE 20

In the last lecture, we saw:

PRIMAL - DUAL ALGORITHM FOR MINIMUM SPANNING TREE:

- Assume $y_\pi^0 = 0 \quad \forall \pi$. Notice that we do not want to explicitly write this out since there are exponentially many of them.
- Iterative step: At the i^{th} iteration: let E_i denote the set of edges that are tight for y^i.
  - Find the connected components of $G_i = (V, E_i)$. Let π_i be the partition given by these components.
  - Increase $y^i_{\pi_i}$ until some edge becomes tight.
- Termination: Previous step terminates when the tight edges form one connected component.

Today: Proof of correctness of this algorithm.

Reference: Lectures 20, 21 of Prof. Sundar's notes.

Recall we are given an undirected graph:

$$G = (V, E) \quad \text{with}$$

cost function $c : E \longmapsto \mathbb{N}_{>0}$

↳ strictly positive integer costs

Assumptions: - G is connected
- no two edges have the same cost

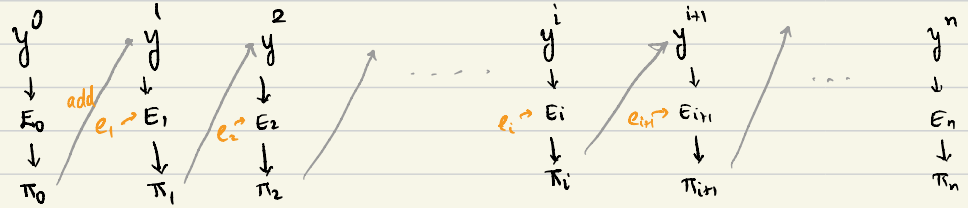
- We made use of the following theorem:

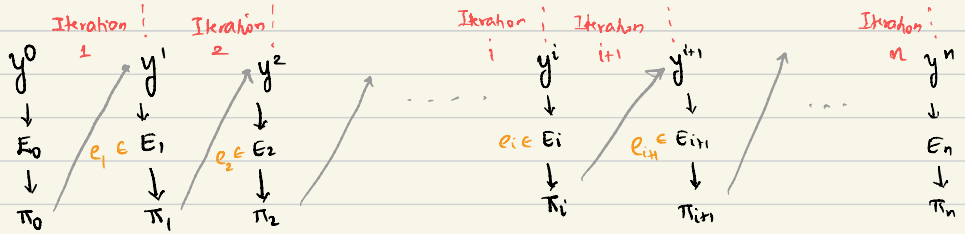
Theorem: A graph is connected

iff

for every partition π, the number of edges that cross the partition is at least $|\pi| - 1$.

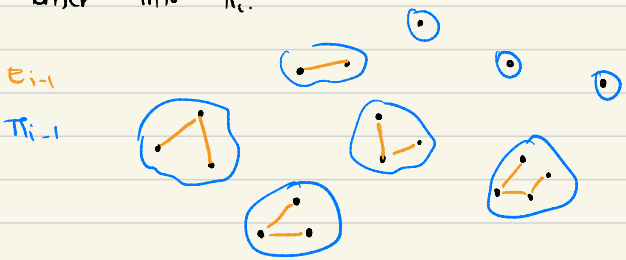
The primal-dual algorithm iteratively generates the following:



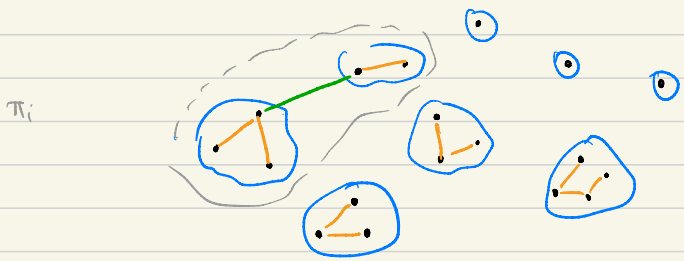


Observation 1: Partition π_i merges parts of π_{i-1}, for all $i \in \{1, \dots n\}$

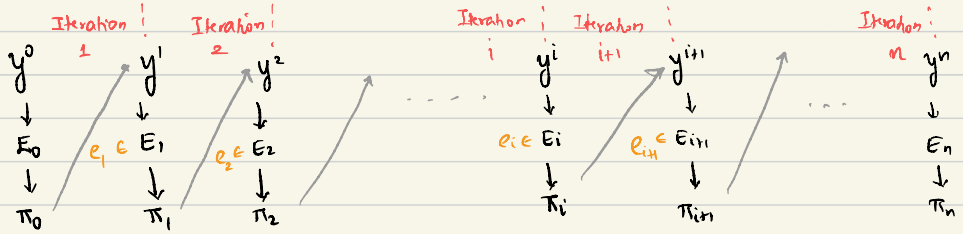
In the i^{th} iteration of our algorithm, we work with partition π_{i-1}. Some edges that cross π_{i-1} become tight and enter into π_i.



The figure above illustrates the edges in E_{i-1} and the partition π_{i-1}



When a cross edge is added to this graph, two parts merge into one connected component, hence one part of π_i.



Observation 2: Exactly one edge e_i is added at iteration i.

At iteration i, the algorithm considers partition π_{i-1} which is the connected components of the graph $G_{i-1} = (V, E_{i-1})$ restricted to edges that have been made tight so far.

Let C_{i-1} be the edges that cross π_{i-1}. Notice that each edge that crosses π_{i-1} also crosses $\pi_{i-2}, \pi_{i-3}, \dots, \pi_1, \pi_0$, due to Observation 1.

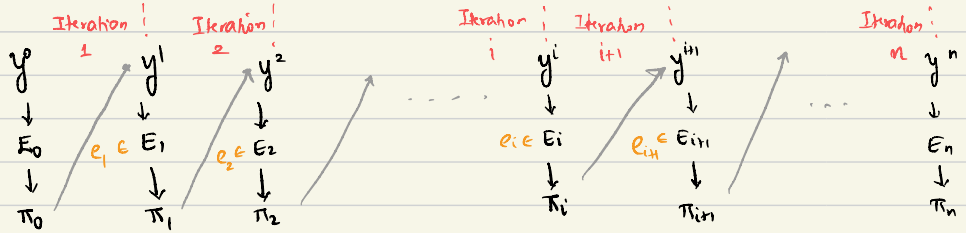
$\therefore$ The dual constraint for each $e \in C_{i-1}$ looks like: $\sum_{e \text{ crosses } \pi} y_\pi \leq c_e$

$$y^{i-1}_{\pi_0} + y^{i-1}_{\pi_1} + \dots + y^{i-1}_{\pi_{i-2}} + y^{i-1}_{\pi_{i-1}} + (\text{some other } y^{i-1}_{\pi'\text{s}} \text{ that are } 0) \leq c_e$$

We also have $y^{i-1}_{\pi_{i-1}} = 0$, and $y^{i-1}_j > 0$ for $j \in \{1, \dots i-2\}$

$\therefore$ To get y^i, the value of $y^{i-1}_{\pi_{i-1}}$ can be increased till the edge with the least weight in C_{i-1} becomes tight.

Due to our assumption on the graph, every edge has a different weight. Therefore exactly one edge becomes tight and this gets into E_i.



Observation 3: $|\pi_i| = |\pi_{i-1}| - 1 \quad \forall\, i \in \{1, \dots n\}$

We have seen in Observation 1 that adding a cross edge of π_{i-1} merges two parts of π_{i-1} into 1.

From observation 2, exactly one edge is added in iteration i.

$\therefore$ Two parts of π_{i-1} become one part in π_i.

Hence $|\pi_i| = |\pi_{i-1}| - 1$

Corollary 1: $|\pi_0| = |V|$

$|\pi_1| = |V| - 1$

$\vdots$

$|\pi_i| = |V| - i$

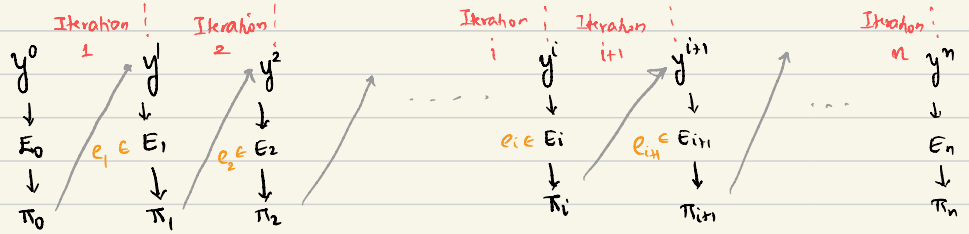
$|\pi_n| = |V| - n$

Observation 4: $n = |V| - 1$

At termination, we have a partition π_n with a single component.

$\therefore |\pi_n| = 1$. From above corollary, $|V| - n = 1$,

$\therefore n = |V| - 1$.



Observation 5: $c_{e_1} + c_{e_2} + \dots + c_{e_n}$

$$= (|\pi_0| - 1)\, y^n_{\pi_0} + (|\pi_1| - 1)\, y^n_{\pi_1} + \dots + (|\pi_{n-1}| - 1)\, y^n_{\pi_{n-1}}$$

The dual constraints that are tight for y^n correspond to the edges $E_n = \{e_1, e_2, \dots, e_n\}$.

$\therefore$ We have $\sum_{e \text{ crosses } \pi} y^n_\pi = c_e \quad \forall e \in \{e_1, e_2, \dots, e_n\}$

Each e_i crosses $\pi_0, \pi_1, \dots \pi_{i-1}$ and not $\pi_i, \dots \pi_n$.
Notice that $y^n_\pi > 0$ only if π is one of $\pi_1, \dots, \pi_n$.

$\therefore$ We get:

$$y^n_{\pi_0} = c_{e_1}$$
$$y^n_{\pi_0} + y^n_{\pi_1} = c_{e_2}$$
$$\vdots$$
$$y^n_{\pi_0} + y^n_{\pi_1} + \dots + y^n_{\pi_{n-1}} = c_{e_n}$$

$$\therefore n \cdot y^n_{\pi_0} + (n-1)\, y^n_{\pi_1} + \dots + y^n_{\pi_{n-1}} = c_{e_1} + \dots + c_{e_n}$$

$n = |V| - 1$
$|\pi_0| = |V|$
$n = |\pi_0| - 1$
$|\pi_i| = |V| - i = n + 1 - i$
$|\pi_i| - 1 = n - i$

From Observation 4 & Corollary 1, we get the required result.

Recall:

PRIMAL:

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

$$\sum_{e \text{ crosses } \pi} x_e \geq |\pi| - 1 \quad \forall \pi$$

$$x_e \geq 0$$

DUAL:

$$\max \sum_{\pi} (|\pi| - 1) y_\pi$$

$$\sum_{e \text{ crosses } \pi} y_\pi \leq c_e \quad \forall e$$

$$y_\pi \geq 0 \quad \forall \pi$$

Let $y = y^n$ the dual soln. obtained at the end of the algo.

Define a primal soln. x as:

$$x_e = 1 \quad \text{if } e \in E_n$$
$$= 0 \quad \text{otherwise}$$

Claim: x is a feasible soln. to primal

The graph $G_n = (V, E_n)$ is connected, by the termination condition of the algo.

$\therefore$ for every partition π of V,

the number of cross edges in G_n is atleast $|\pi| - 1$.

This implies x is a feasible soln. for primal.

PRIMAL:

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

$$\sum_{e \text{ crosses } \pi} x_e \geq |\pi| - 1 \quad \forall \pi$$

$$x_e \geq 0$$

DUAL:

$$\max \sum_{\pi} (|\pi| - 1) y_\pi$$

$$\sum_{e \text{ crosses } \pi} y_\pi \leq c_e \quad \forall e$$

$$y_\pi \geq 0 \quad \forall \pi$$

We will now show that x and y are optima of primal & dual respectively.

Cost of primal at $x = \sum_{e \in E} c_e x_e$

$= c_{e_1} + c_{e_2} + \cdots + c_{e_n}$ where $E_n = \{e_1, \ldots e_n\}$

$= (|\pi_0| - 1) y_{\pi_0} + (|\pi_1| - 1) y_{\pi_1} + \cdots + (|\pi_{n-1}| - 1) y_{\pi_{n-1}}$

(from Observation 5)

$=$ Cost of dual at y.

This implies x and y are optima of primal and dual respectively.

↳ We have seen in the lecture on complementary slackness that given feasible solns. x_0, y_0 of primal & dual: $c^T x_0 = b^T y_0$ iff x_0, y_0 are optima

Notice that the primal soln. is also a soln. of ILP.

In general: LP opt. $\leq$ ILP optimum (as this is minimization)

Since the LP optimum occurs at an integral point, it is also an ILP optimum.

– This shows that we have obtained a minimum spanning tree.

Complexity: The algorithm maintains the partitions.

- At each iteration, the algo finds the least weighted edge among all cross edges. This can be done in $O(|E|)$.
- It merges two parts due to the addition of this edge. This is also a $O(|V|$ procedure.
- The number of iterations is $O(|V|$.

Hence overall, $O(|V|\,(|V|+|E|))$

This naive complexity analysis already puts it in polynomial-time.

Summary:

- A primal-dual algorithm solving MST.
- Mimicks Kruskal's algo
- Proof of correctness using strong duality.