Maximum Spanning Tree and Minimum Cost Arborescence

CS 602

Peram Ankshita
Sirigudi Meghana
Vishwesh Kandukuri

November 1, 2023

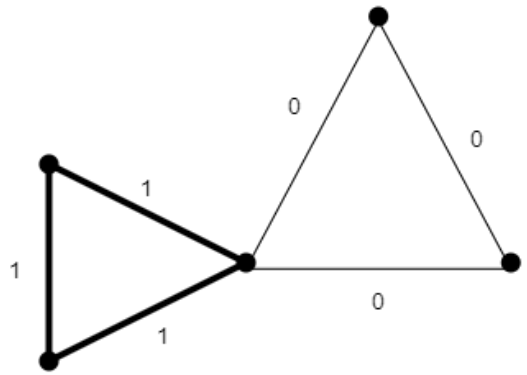
Outline

- Applications of duality to below problems:
  - Finding maximum spanning trees (MST)
  - Finding minimum cost arborescences

Maximum Spanning Tree(MST)

Given a graph $G = (V, E)$, and edge weights $w_e \geq 0$, the goal is to output a spanning tree of maximum weight

- Consider variable $\{x_e\}_{e \in E}$
- For a set $S \subseteq V$, we denote by δS the set of edges leaving S
- For $A \subseteq E$, define $x(A) = \sum_{e \in A} x_e$.
- Example:



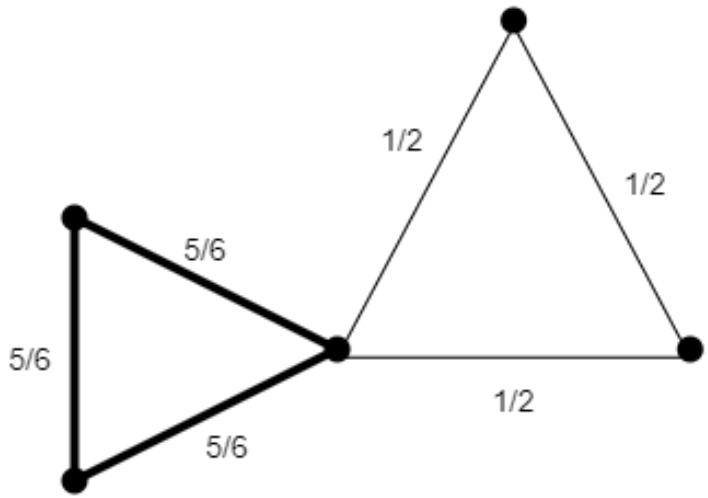
LP for MST

Consider the following LP:

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & 1 \geq x_e \geq 0 \\ & \sum_{e \in E} x_i = n - 1 \\ & x(\delta S) \geq 1 \quad \forall S \neq \varnothing, V \end{aligned}$$

Correctness of LP

- Is every vertex of above LP integral ?
- Consider this example,



Check..

- The maximum weight spanning tree has weight 2
- However, the LP solution given here (with $x_e = 1/2$ on the thin edges, and $x_e = 5/6$ on the thick ones) has $w^\top x = 3 \cdot 5/6 = 2.5$
- It also has $\sum_e x_e = 3 \cdot 1/2 + 3 \cdot 5/6 = |V| - 1$, and you can check it satisfies the cut condition
- **Thus it is clear that above LP doesn't give desired solution!!**

Trying in another way..

- Understand this LP:
  - For $S \subseteq V$, let E_S denote all edges between vertices in S. (For simplicity, we will assume in this lecture that all the edge weights are non-negative.)

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & x(E_S) \leq |S| - 1 \qquad \forall S \subseteq V, |S| \geq 1 \\ & x_e \geq 0 \end{aligned}$$

Remarks

Remark Any spanning tree satisfies these constraints. Therefore, opt(P) is at least the weight of the maximum spanning tree.

- Recall that **Kruskal's algorithm** starts with a forest consisting of all the vertices, and iteratively adds the heaviest edge which connects two trees.

Example

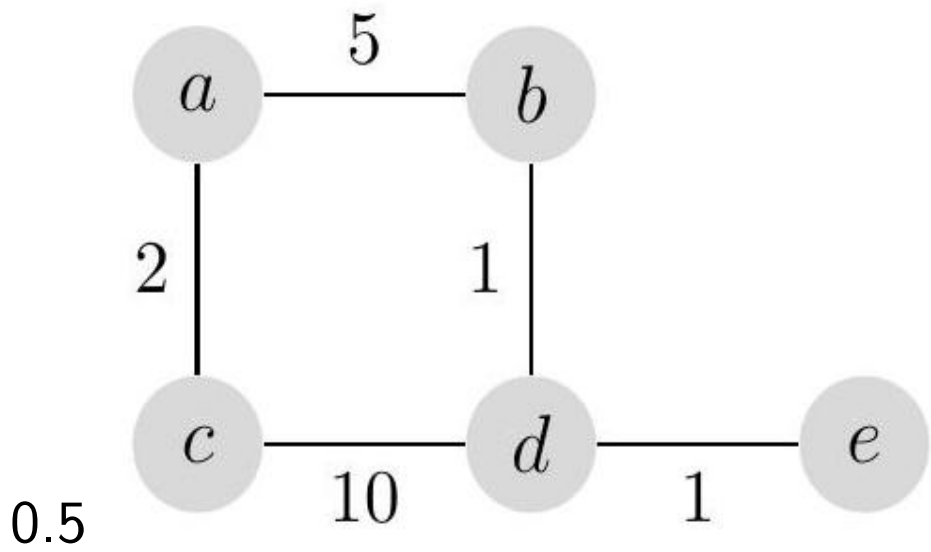


Figure: Graph G

Figure: Example for Kruskal's algorithm

Here,in the above graph G according to Kruskal's algorithm,first we choose 'cd' edge since it has highest weight and then 'ab',followed by 'ac' and 'de' to avoid cycle.Thus if we iterate maximum spanning tree would be e-d-c-a-b with maximum weight=18.

Primal Solution

Kruskal: Pick edges $K = \{e_1, e_2, \ldots, e_{n-1}\}$ with $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_{n-1})$. Then a primal solution is given by

$$x_e = \begin{cases} 1 & e \in K \\ 0 & e \notin K \end{cases}$$

The value of this solution is $\sum_e w_e x_e$, which is exactly the value of the Kruskal solution.

Claim

Theorem: There exists an integer optimum for the LP P, and Kruskal's algorithm finds it

- Proof. We will construct a dual solution such that its value is the value of the MST which Kruskal finds.

Dual LP

For a set S, write $r(S) := |S| - 1$. (This is the size of a spanning tree on S.)

$$\begin{aligned} \min \quad & \sum_{S \neq \emptyset} r(S) y_S \\ \text{s.t.} \quad & \sum_{S : e \in E_S} y_S \geq w_e \forall e \in E \\ y_S \geq 0 & \end{aligned}$$

Dual Solution

Suppose that we run Kruskal on our graph. We consider the sequence of components satisfied by the addition of each edge. This naturally induces a tree structure on the edges of the MST, where the parent of a subtree corresponding to some component C is the first edge added to the MST which leaves C

Example

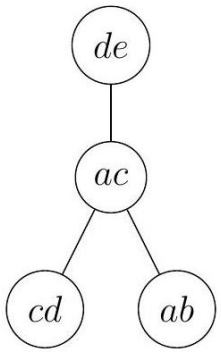


Figure: Fig b:Resulting Tree

Dual Solution

We will consider the tree induced by Kruskal's algorithm.
Let $V(e_j)$ be the set of vertices spanned by the edges in the subtree rooted at e_j.
Define a dual solution y_S by

$$y_S = \begin{cases} w_{e_j} - w_{\text{parent}(e_j)} & S = V(e_j) \text{ for some } j \\ 0 & \text{else} \end{cases}$$

For Figure b, we have $y_{\{c,d\}} = 10 - 2 = 8, y_{\{a,b\}} = 5 - 2 = 3, y_{\{a,b,c,d\}} = 2 - 1 = 1$, and $y_{\{a,b,c,d,e\}} = 1 - 0 = 1$.

Remark

Lemma: y_S is feasible.

$$\sum_{S:e\in E_S} y_S \geq \sum_i y_{V(p_i)} = w_{p_1} - w_{p_2} + \cdots + w_{p_k} = w_{p_1} = w_{e_j} \geq w_e$$

Remark

Lemma:

$$\sum_{S \neq \varnothing} r(S) y_S = \sum_{e \in K} w_e$$

(Recall K is the spanning tree output by Kruskal.)

- We prove by (strong) induction the slightly stronger statement that

$$\sum_{S \subseteq V(e_j)} r(S) y_S = \sum_{e \in T(e_j)} w_e - r\left(V\left(e_j\right)\right) w_{\text{parent of } e_j}$$

  for every $e_j \in K$.
- Thus the maximum spanning tree LP has an integer optimum given by Kruskal's algorithm.

Minimum Cost Arborescence

Think of these as spanning trees on directed graphs. Given a directed graph $G = (V, E)$ with a root vertex r, an arborescence of G is a subgraph $T = (V, E_T)$ such that:

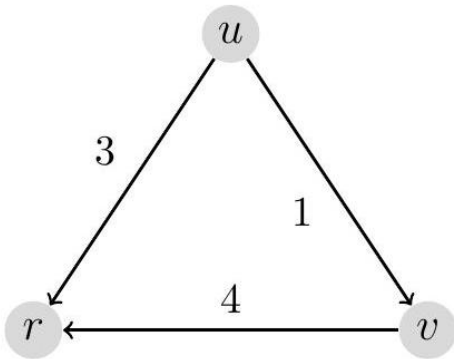
- Every node is connected to r, and there are no cycles even if we ignore directions.
- Every node has a directed path to r.

Remarks

- One often sees this definition with directed paths from r to other nodes, but we will use this convention.
- An arborescence may not exist for a given root r, but a certificate of infeasibility is a vertex with no path to r.

Remarks

Note that it may also not be unique. Furthermore, the following example shows that adapting Prim's algorithm (greedily starting at the root) may not yield an optimal solution.



Primal-Dual Algorithm

- Notation: We write $\delta^+ S$ to denote the set of edges leaving S for any $S \subseteq V$
- Primal:
  We will assume $c_e \geq 0$ for every e. The primal LP is

$$\begin{aligned} \min & \sum_{e \in E} c_e x_e \\ \text{s.t. } & x\left(\delta^+ S\right) \geq 1 \quad S \text{ valid} \\ & x_e \geq 0 \end{aligned}$$

- where we will call $S \subseteq V$ valid if S is nonempty and $r \notin S$.

Primal-Dual Algorithm

- Dual:

$$\begin{array}{ll} \max & \sum_{S \text{ valid}} y_S \\ \text{s.t.} & \sum_{S: e \in \delta^+ S} y_S \leq c_e \quad \forall e \\ & y_s \geq 0 \end{array}$$

- where we will call $S \subseteq V$ valid if S is nonempty and $r \notin S$.

Primal-Dual Algorithm

- If zero-weight edges connect every $v \neq r$ to r, then we get a (integral) primal solution of value 0 (using depth first search or similar). A matching dual solution sets $y_S = 0$ for every S. In particular, this is optimal.

Primal-Dual Algorithm

- Otherwise, consider the graph restricted to zero-weight edges. Choose a maximal strongly connected component C of this subgraph. Then in the graph with all edges, there are no zero-weight edges out of C. Let $c^* = \min_{e \in \delta^+ C} c_e$. For each $e \in \delta^+ C$, define updated weights $c'_e = c_e - c^*$.
  Run the algorithm recursively on G with C contracted to one vertex $\hat{C}$ and with the updated weights to get optimal primal/dual solutions x_e, y_S for the contracted graph. Inductively, x_e will be integral.
  Let A be an arborescence on C (of zero cost). Define primal/dual solutions $\hat{x}_e, \hat{y}_S$ for the uncontracted graph by

$$\hat{x}_e = \begin{cases} x_e & e \notin C \\ 1 & e \in A \\ 0 & e \in C \backslash A \end{cases} \qquad \hat{y}_S = \begin{cases} y_S & C \not\subset S \\ y_{\hat{C}} + c^* & S = C \\ y_{\{S \backslash C\} \cup \{\hat{C}\}} & C \subsetneq S \end{cases}$$

Remarks

- If $\sum c'_e x_e = \sum y_S$ (i.e., if the primal and dual solutions mutually certify optimality), then $\sum c_e \hat{x}_e = \sum \hat{y}_S$. This holds since $\sum y_S + c^* = \sum \hat{y}_S$. Furthermore, in any minimal arborescence on the contracted graph, exactly one edge from $\delta^+ C$ will be chosen.

Lemma

- $\hat{x}_e$ and $\hat{y}_S$ are feasible.
  Proof: $\hat{x}_e$ is feasible because x_e is feasible (and clearly the arborescence A satisfies the conditions). To show $\hat{y}_S$ is feasible, we only need to check $\sum_{S:e\in\delta^+S} \hat{y}_S \leq c_e$ for $e \in \delta^+ C$. It is easy to see that this holds by definition of $\hat{y}_S$, since $\sum y_S \leq c_e - c^*$.

Conclusion

This LP we wrote for arborescences is very similar to the one we first wrote for spanning trees, but that one did not work, whereas this one does! Interesting. Indeed, this LP can be used to give an algorithm for MSTs on undirected graphs.
Indeed, take an undirected graph and replace each undirected edge by two directed edges of the same weight, pointing in opposite directions. Now the max-weight arborescence in this digraph has the same weight as the maximum spanning tree in the original graph. So an LP that looks pretty similar to the failed undirected one (namely $\max\left\{w^\top x \mid x(\partial v) = 1, x(\partial S) \geq 1, x \geq 0\right\}$) on that specially constructed directed graph gives an arborescence that corresponds to an integral maximum spanning tree on the original undirected graph.