~~PS8: Tarjan's $O(m\,\alpha(m,n))$ time algorithm to do MST verification.~~ 1

Lecture #3: Min Cost Arborescence. / LP methods.

We saw MSTs on undirected graphs. In HW1 we saw that these are a special case of matroids. Today we'll study spanning trees in digraphs. These are slightly different, need a more careful algorithm than just a greedy algorithm.

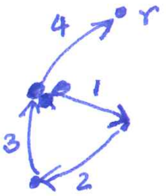
Def: Given a digraph $G=(V,A)$, and a root vertex $r \in V$, an r-arborescence is a set of arcs $B \subseteq A$ s.t. (1) the outdegree of each vertex (except r) is 1 and (2) the graph (V,B) has a path from each x to r.

Can we find if G has an r-arborescence? Sure, reverse arcs and run DFS from root.

Given arc weights w_a, find a min cost r-arborescence.

exercise: ~~distinct wts~~
{ HW: rooted vs unrooted
: branchings vs arborescence.
: min cost vs max cost.

Attempt 1: greedy. fails. Ex:



[More sophisticated versions? don't see what]

[Assume: wts are non-neg, else add M to each.

[Assume: r has no outgoing arcs. Don't need it.

[~~Assume: wts of all arcs out of a node are distinct.~~

So here's a try: for each node v, look at out arcs and let $\min wt(v)$ = min wt of outgoing arc.

$\forall v$, Reduce wts of all arcs out of v by $\min wt(v)$.
N.b. weights remain non-negative.

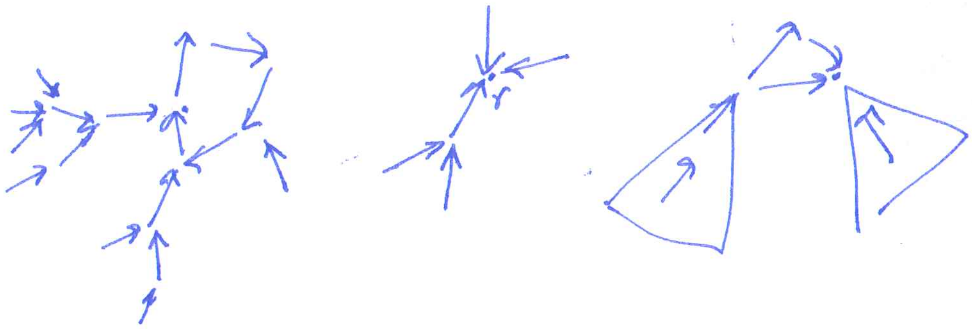
Claim: T^* is min wt arbor in G' $\iff$ T^* is MWA in G.

So each node has - a 0-wt arc out of it ← if multiple, pick one. ~~because due to distinct wts assumption~~

- only non-negative wts.

<u>Fact</u>: if $\exists$ a 0-cost r-arborescence, we're done. [OPTimum].

So sps not. Then 0-cost graph looks like

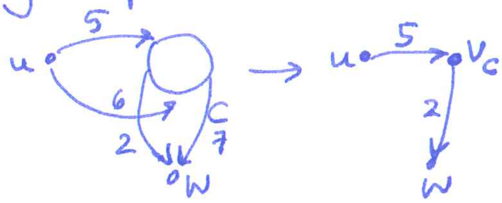


each "sink" component is a cycle or r.

Consider any 0-cost cycle C. let $G' = G/C$ i.e. C is contracted into single node called v_C. replace ~~each~~ parallel arcs by cheapest arc.

<u>Claim</u>:

OPT(G') has same cost as OPT(G).

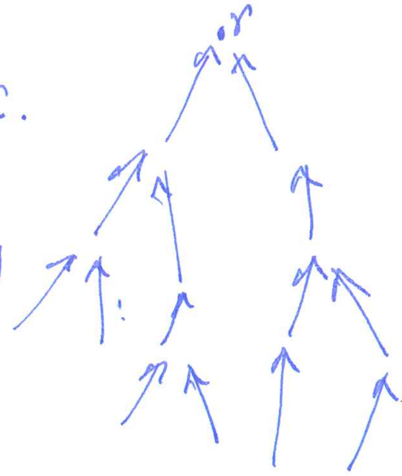


etc.

<u>Pf</u> ($\Rightarrow$) OPT(G') $\leq$ OPT(G).

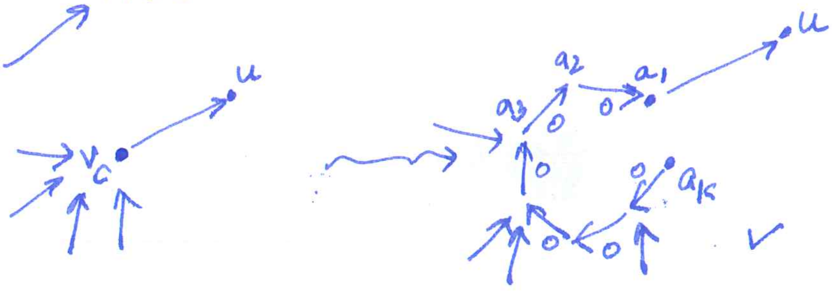
Just take OPT(G). Contract all nodes in C. into v_C.

drop all but one arc out of v_C, cost only decreases. ✓



($\Leftarrow$) OPT(G) $\leq$ OPT(G').

Consider T'



✓

In fact, this says. find a min cost soln in G' (3)

and this gives an explicit algorithmic way to extend the soln to G. with same cost.

By induction solve on G'..

Note: G' is smaller graph. so induction is valid.

Runtime: $O(mn)$.

due to Edmonds [67]
Bock [71]
Chin-Liu [65].

[Can we do faster?]

Yes: Gabow Galil Spencer Tarjan give $O(m + n\lg n)$. '86

Don't think anything better known.

One way of looking at solution: Each vertex must pay for cost of edge out of it.

So each vertex pays $\min wt(v)$. $\leftarrow$ P_v or $P_{\{v\}}$ to reduce costs to 0.

But not enough to escape cycle, say. So cycle forms a coalition C.

Now jointly they pay the cost of cheapest edge out of the cycle. say P_C. to reduce some edge cost to 0. etc....

Properties: $\left[\begin{array}{l} P_S \geq 0 \quad \forall S \subseteq V. \\ \sum_{S: e \in \partial^+ S} P_S \leq y_e. \quad \forall \text{edges} \end{array}\right]$ "valid" prices.

$\partial^+ S$ = set of arcs leaving Set S.

Prices and Arborescence B are in equilibrium if

1. prices are valid
2. $\forall$ arcs $a \in B$, $w_a \stackrel{=}{\leq} \sum_{S: a \in \partial^+ S} P_S$ equality $\leftarrow$ we can use an arc only if it is fully paid for.
3. $\forall S$ st $P_S > 0$, $\exists$ single arc in $\partial^+ S \cap B$.

Thm: if $\exists$ valid prices and $\overset{\swarrow \text{sol}^n}{B}$ in equilibrium $\Rightarrow$ B is optimal

Pf: $\sum_{e\in B} w_e \overset{\text{PP} \S 2.}{=} \sum_{e\in B} \sum_{S: e\in\partial^+ S} p_S = \sum_S p_S (\# B \cap \partial^+ S)$

$\underset{\#B\cap\partial^+S = 1}{=} \sum_S p_S.$

$\forall B^*, \sum_{e\in B^*} w_e \geq \sum_{e\in B^*} \sum = \sum_S p_S(\# B^* \cap \partial^+ S)$

$\#B^*\cap\partial^+S \geq 1$ because sol^n.

$\geq \sum_S p_S.$ ∎

Fact: Our solution maintains prices and B in equilibrium.

Check: initially $p_v > 0$ only on singletons. And sol^n has out-degree $= 1$. use only 0-cost edges.

Moreover, inductively

~~value~~ maybe $p_{v_c} > 0$ and $p_{S'} > 0$ for $S' \ni v_c$

then extend to p_C and $p_{(S' \setminus v_c) \cup C}$ and $p_x \ \forall x \in C$.

All have one edge crossing these sets

Interesting but mysterious (perhaps). Here's another way of looking at it. Via LPs.

$$\min \sum w_a x_a. \qquad (P)$$

$$\text{st} \quad x(\partial^+ S) \geq 1 \quad \forall S \subseteq V \setminus \{r\}.$$

$$x \geq 0.$$

$$\max \sum_S p_S \qquad (D)$$

$$\text{st} \quad \sum_{S: a\in\partial^+ S} p_S \leq w_a.$$

$$p_S \geq 0.$$

This is just a primal dual pair of LPs. ← (may be fractional!)

Suppose we find a solution to the primal which is integral.

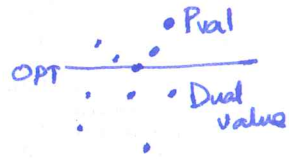
call it $x^* \in \{0,1\}$ st x satisfies constraints of (P).

And we find a way to set the dual variables p_S^*

st p^* satisfies constraints of the dual (D).

Thm: [weak duality] $\forall\ x^* \in SOL(P)$ $p \in SOL(D)$

then $\sum w_a x_a^* \geq \sum_S p_S$.



~~Thm: St~~ Corollary: if $\exists\ x^*, p^*$ sol^{n}s s.t. $\sum_a w_a x_a^* = \sum_S p_S^*$

then x_a^* and p_S^* are both optimal sol^{n}s.

Thm: [complementary slackness] p, x are feasible sol^{n}s st.

sps (a) $p_S > 0 \Rightarrow x(\partial^+ S) = 1$

(b) $x_a > 0 \Rightarrow \sum_{S: a \in \partial^+ S} p_S = w_a$.

⇒ both are optimal solutions

Pf: $\sum_a w_a x_a = \sum_a \underbrace{\Big(\sum_{S: a\in\partial^+ S} p_S\Big)}_{\text{by (b)}} x_a = \sum_S p_S \underbrace{\Big(\sum_{a \in \partial^+ S} x_a\Big)}_{1 \text{ by (a)}} = \sum_S p_S$

now use Corollary ∎.

So this gives another way of arguing that our algorithm was optimal.

On HW 2, show that $\exists$ a setting of dual variables to prove that matroid greedy algorithm is optimal.
