CS6381 Lecture 22 : 10·23-2020

Odds and Ends. (=Promises made in CS6363)

1. Directed Spanning Tree Problem: (see also Tardos, Kleinberg)

Let G be a dir. graph $G = [V; E]$ with edge weights $w(e)$ $e \in E$.

Also, $r \in V$: root of the dir. spanning tree.

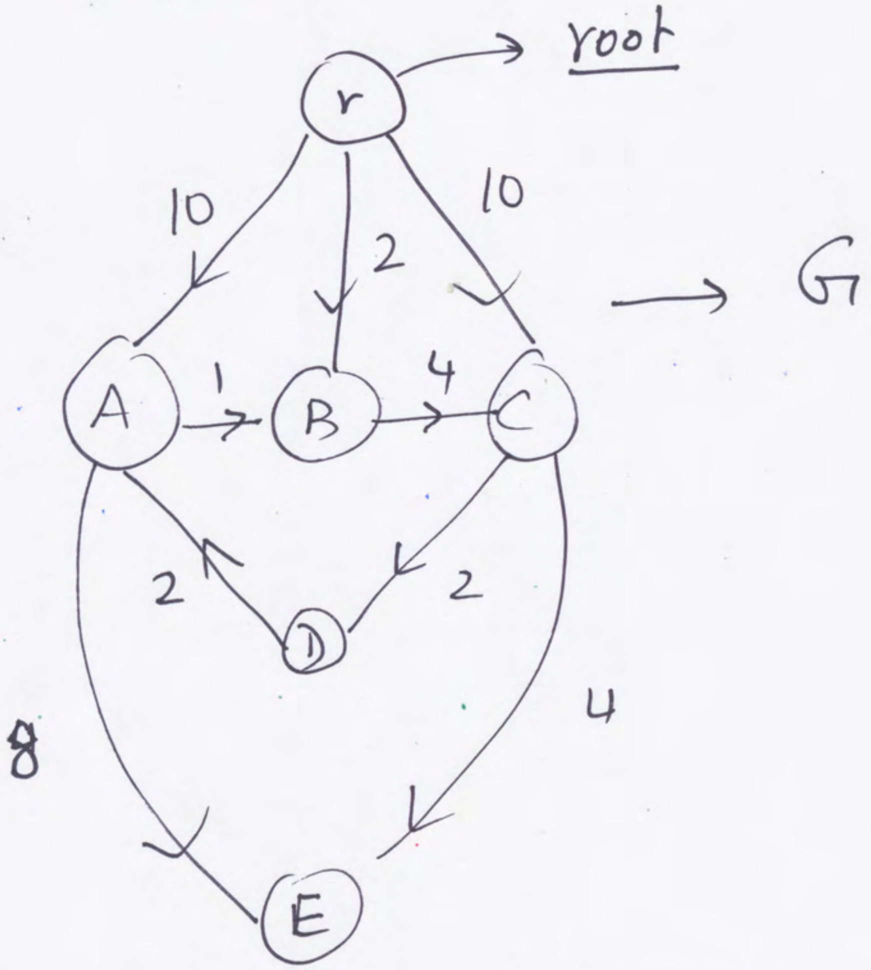
($\exists$ a dir. path in the tree from r to every other node; and it is sp. tree when dir. are ignored)

Want: a rooted (at r) dir. sp. tree with min total weight.

[This was also solved by J. Edmonds as a special case of something much bigger]. But we present a "much easier" version. It also involves "Shrinking". And in some sense, it is a "greedy" algorithm as well.

We describe the algorithm via an example (this will be general result — for now believe me!)

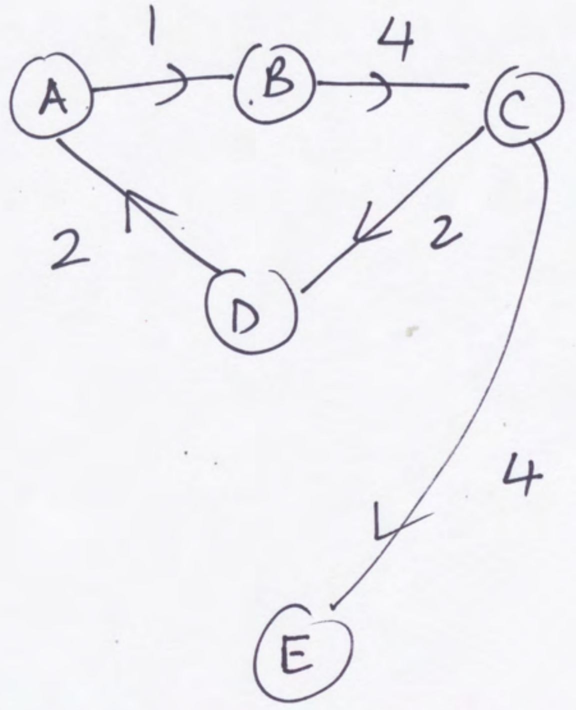
<u>Example</u>. We assume $\exists$ a dir. path from r to each other node in G. [Moreover, we may assume w/o loss, $\exists$ no edges going into r – these can be removed – they will not be part of <u>any</u> rooted (at r) dir. sp. tree in G].



Every dir. sp. tree has $|V|-1$ edges; moreover in such trees, each node $\neq r$ has <u>one</u> edge coming into it. But not all subsets satisfying these two conditions is a dir. sp. tree.

Since each node has exactly one edge coming into it, a greedy approach will be to select cheapest edge into each node.

For our example, it would be:



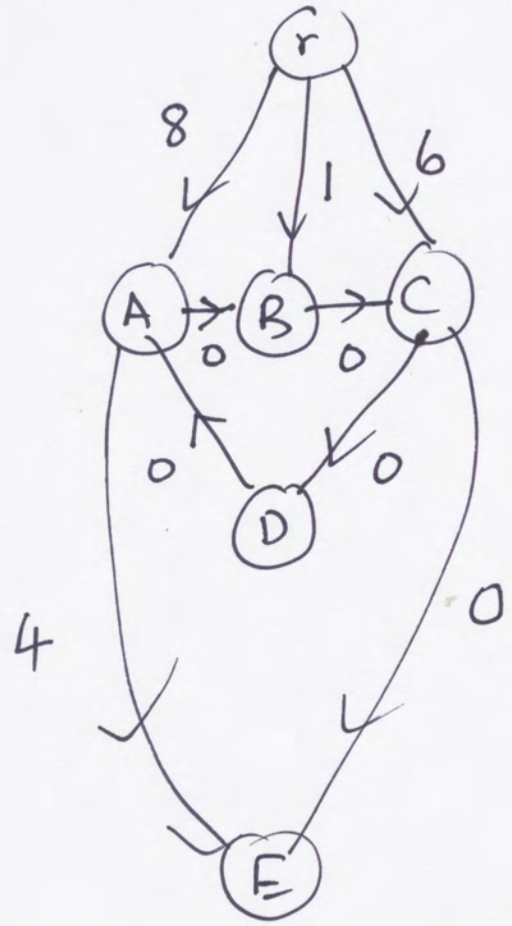
This is "obviously" not a dir. sp. tree.
It has dir. cycles.

Recall: If we subtract/add a constant for each edge going into a node, problem solution does not change (although total weight will)

We do this to get some clarity.

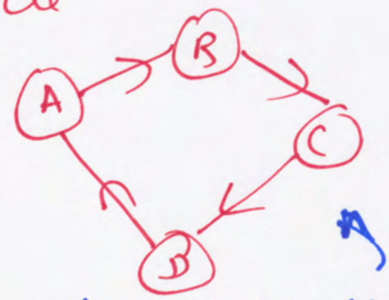
A : 2 ; B : 1, C : 4, D : 2, E : 4

to get:



Edges with weight 0 here, were chosen by the first step of our greedy algorithm

But it produced a cycle

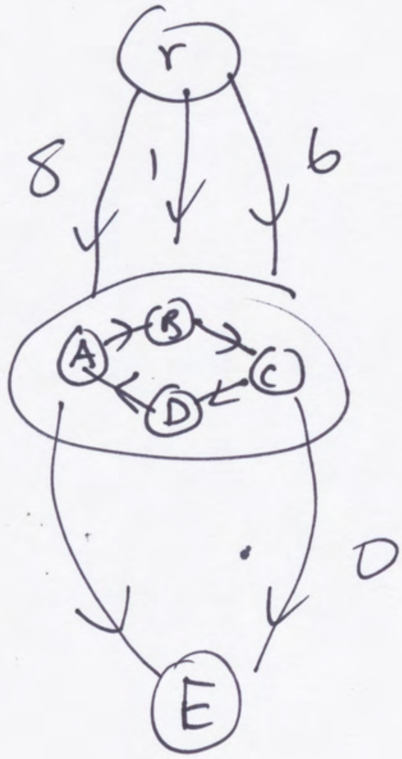


We now "Shrink" this

Basis

For any subset S of nodes such that $r \neq S$,
and any dir. sp. tree rooted at r,
there must be at least one edge entering this set.

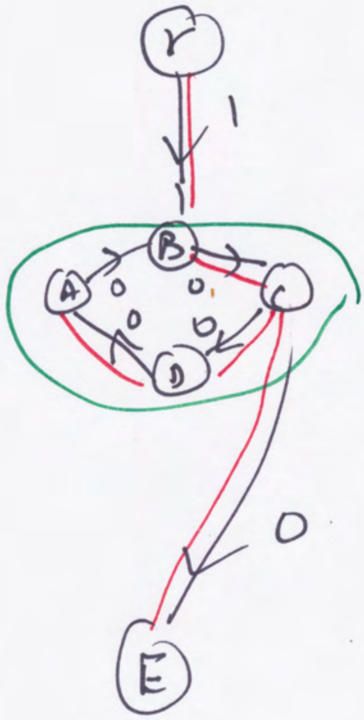
Shrunken Graph (edge costs modified)



Use greedy algorithm
On this Shrunken
Graph.

(ABCD) : 1

E : 0



Since ∃ (r,B) going
into B, remove
(AB) to get a
dir. sp. tree

Ex: Prove this is optimal solution.

Of course, shrinking may be done many times
as also "Unshrinking"!

Ratio Optimization in Combinatorics:

(See: N. Megiddo; R.M. Karp, Maffioli, RC) See also Dinkelbach & Jagannathan

Ratio Spanning Trees: (Maffioli, RC)

G: undir. connected $= [V; E]$; two numbers for each edge $a(e), b(e)$; $b(e) > 0\ \forall e \in E$

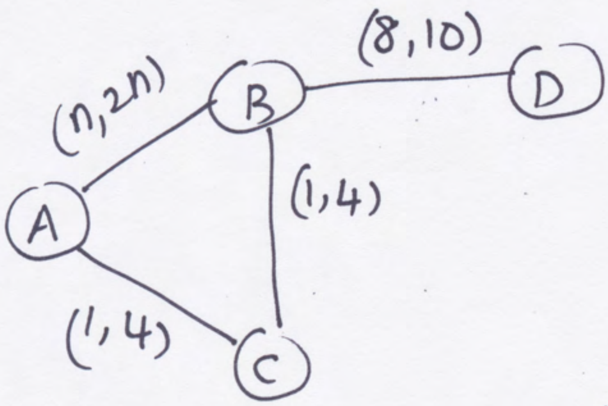
Want: Sp. tree T that $\min\limits_{T} \dfrac{\sum_{e\in T} a(e)}{\sum_{e\in T} b(e)}$.

An Example that shows that a potential greedy algorithm Candidate does not Work.

Suppose we order edges so that $\dfrac{a(e_1)}{b(e_1)} \le \dfrac{a(e_2)}{b(e_2)} \le \cdots$

and ~~select~~ include edges in that order unless the new edge creates a cycle (in which case we decide not include that edge).

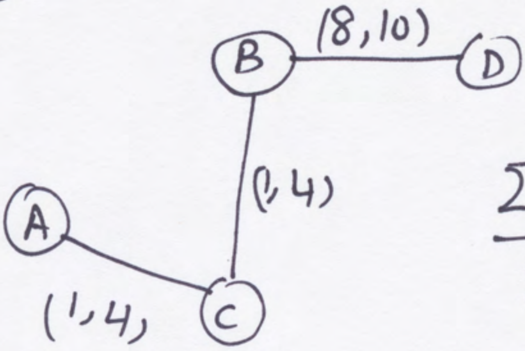
Consider the example on the next page to show that this does not always produce the desired result.



(a, b):

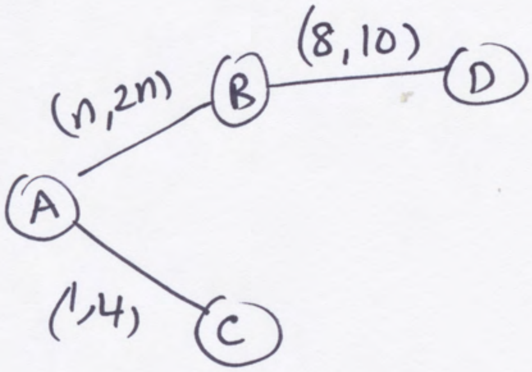
<u>n: large</u>

GA:

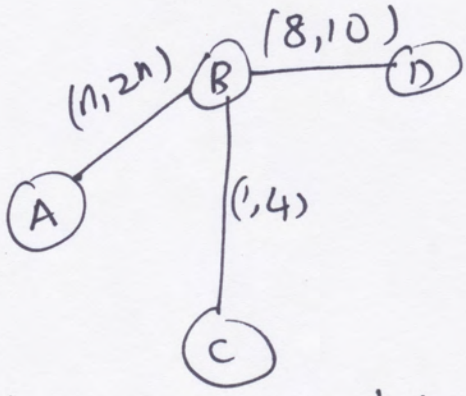


$$\frac{\sum a(e)}{\sum b(e)} = \frac{10}{18} = \frac{5}{9} > \frac{1}{2}$$

<u>OPT</u>



<u>One possibility</u>



<u>Other possibility</u>

$$\frac{\sum a(e)}{\sum b(e)} = \frac{n+9}{2n+14} \xrightarrow[n\to\infty]{} \frac{1}{2}$$

Recall that only operation used in <u>all</u> algorithms for finding MST is <u>Comparison</u>. So, once an total ordering of the edges is given, MST is uniquely determined. ~~(assuming no~~ We use this <u>and</u> the process of defining a family of regular MST problems to solve the ratio MST problem using a procedure which is due to W. Dinkelbach & R. Jagannathan for general ratio optimization.

Given $a(e), b(e)$; $e \in E$, let us define a family of problems with: ~~$w(e)$ $w(e,\lambda)=$~~

$$w(e,\lambda) = a(e) - \lambda b(e) \quad e \in E$$

Now let $T^*(\lambda)$ be a min. sp. tree on G with weights $w(e,\lambda)$; $e \in E$.

(<u>Parametric MST</u>)

We discuss some properties of parametric MST in relation to Ratio MST.

Let $w_T(\lambda) = \sum_{e \in T} w(e, \lambda)$

and let $w^*(\lambda) = \min_{T} w_T(\lambda) = \overset{w_{T^*(\lambda)}(\lambda)}{\cancel{w(T^*(\lambda), \lambda)}}$

$T \rightarrow$ over all sp. trees of G.

If $\lambda > \lambda'$, $w_T(\lambda) < w_T(\lambda') \; \forall T$

since $b(e) > 0 \; \forall e \in E$.

$\therefore$ ~~Let T^*~~

$$w^*(\lambda') = \underbrace{w_{T^*(\lambda')}(\lambda')}_{\text{best tree for } \lambda'} \overbrace{> \underbrace{w_{T^*(\lambda')}(\lambda)}_{\text{Some tree for } \lambda}}^{\text{Same tree, for } \underline{\lambda' < \lambda}} \geq \underbrace{w_{T^*(\lambda)}(\lambda)}_{\text{best tree for } \lambda} = w^*(\lambda)$$

$\therefore$ $w^*(\lambda)$ is non-increasing function of λ.

Lemma [Dinkelbach, Jagannathan]

Let $\min_T \left\{ \frac{\sum_{e \in T} a(e)}{\sum_{e \in T} b(e)} \right\} = \lambda^*$; Then

$$w^*(\lambda) = \begin{cases} > 0 & \lambda < \lambda^* \\ = 0 & \lambda = \lambda^* \rightarrow T^*(\lambda) \text{ is opt for RMST.} \\ < 0 & \lambda > \lambda^* \end{cases}$$

Proof :

T^* : opt for RMST

i) $\lambda > \lambda^* = \dfrac{\sum_{e \in T^*} a(e)}{\sum_{e \in T^*} b(e)}$

$\therefore \; w_{T^*}(\lambda) < 0$

$\therefore \; w^*(\lambda) < 0$

ii) $\lambda < \lambda^* = \dfrac{\sum_{e \in T^*} a(e)}{\sum_{e \in T^*} b(e)} \leq \dfrac{\sum_{e \in T} a(e)}{\sum_{e \in T} b(e)} \quad \forall T$

$\therefore \; \dfrac{\sum_{e \in T} a(e)}{\sum_{e \in T} b(e)} > \lambda \quad \forall \lambda$

$\therefore \; w_T(\lambda) > 0 \quad \forall \lambda$

$\therefore \; \underline{w^*(\lambda) > 0}$.

The case $\lambda = \lambda^*$ follows from both of these & $w^*(\lambda)$ is a continuous function.

Recall $\omega(e,\lambda) = a(e) - \lambda\, b(e)$

For fixed λ, finding $T^*(\lambda)$ is MST problem and hence $\exists$ an algorithm that uses only comparisons between $\{\omega(e,\lambda)\}$; i.e

Comparisons of linear functions of λ.

As we change λ, the total orderings of $\{\omega(e,\lambda)\}$ can ONLY CHANGE at point of the type $\omega(e',\lambda) = \omega(e,\lambda)$ $\quad e \neq e'$

i.e $\quad a(e') - \lambda b(e') = a(e) - \lambda\, b(e)$

$$\Rightarrow \underset{\lambda(e,e')}{\overset{\lambda}{=}} = \frac{a(e) - a(e')}{b(e) - b(e')}$$ assuming $\underline{b(e) - b(e') \neq 0}$

$\downarrow$

Only $O(|E|^2)$ values

Moreover $\omega^*(\lambda)$ is monotone non-decreasing

Hence $\exists$ a s. poly time algorithm to solve this problem. Best order in N. Megiddo's work

Previous work applies to ALL problems which for the non-ratio optimization have a SP alg. using ONLY Comparisons.

N. Megiddo

Now we turn to those that use additions and comparisons for the linear optimization

This includes. Ratio trees, Ratio cycles, Ratio p. matchings, etc. [Caution: Ratio SP is NP-C]

Suppose $\exists$ an algorithm that uses ~~$O(p(n))$~~ $p(n)$ additions and $q(n)$ Comparisons to solve the problem:

$$\min_{X \in S} \left[a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \right] \qquad \forall\ \underline{a \in R^{n+1}}$$

Where S is a set of Combinatorial objects such as set of spanning trees in a graph, a perfect matching in a Graph etc. X is the indicator vector of feasible solutions.

We assume $p(n)$, $q(n)$ are polynomial functions and work for arbitrary $[a_0, a_1, \ldots, a_n]$.

Theorem (Megiddo).

$\exists$ an algorithm that solves the problem

$$\min_{x \in S} \frac{a_0 + a_1 x_1 + \cdots + a_n x_n}{b_0 + b_1 x_1 + \cdots + b_n x_n} \quad *$$

in time $\cancel{O[p(n)(q(n)+p(n))]}$ $O[q(n)(p(n)+q(n))]$ assuming that denominator is always positive.

We use Dinkelbach-Jagannathan result again:

Let optimal value be λ^*. Consider the parametric problem:

$$\omega(\lambda) = \min_{x \in S} \left\{ (a_0 - \lambda b_0) + (a_1 - \lambda b_1) x_1 + \cdots + (a_n - \lambda b_n) x_n \right\}$$

Since Denominator is > 0 $\forall x \in S$, it follows that $\omega(\lambda)$ is a monotone decreasing function of λ.

Let algorithm A solve the linear problem in $O(p(n))$ Comparisons and $O(q(n))$ additions.

Remark: additions preserve linearity.

Lemma (D, J):

$$\omega(\lambda) = \begin{cases} >0 & \lambda < \lambda^* \\ =0 & \lambda = \lambda^* \\ <0 & \lambda > \lambda^* \end{cases}$$

Proof: Similar to Ratio Sp. Tree Case.

This result lets us know which region to search for λ^* given $\omega(\lambda)$ (rather its sign)

In algorithm A whenever there is a <u>Comparison</u> it is a Comparison of two linear function of λ. These two linear functions intersect at at most one value of λ. If we know $\omega(\lambda)$ at intersection point (or its sign) then we can restrict ourself to one of these regions where the result of the Comparison does <u>not</u> change.

Since there are $O(p(n))$ Comparisons in algorithm A, we need to solve that many $\omega(\lambda)$ problem each of which take $O(p(n))$ Comparisons and $O(q(n))$ addition. Hence, in $O(p(n)(p(n)+q(n)))$ time we can restrict our Search for λ^* to an interval where $\omega(\lambda)$ is linear. Contains

Solving for $w(\lambda) = 0$ in this interval (Solving a linear equation in λ) we get λ^* which in turn gets us $x^* \in S$.

Please be careful when reading N. Megiddo's paper. This can not be used to solve ratio simple shortest path problem as claimed. This is because, in the presence of negative cycles, we can not find shortest simple paths in poly. time.. We can't even detect if there is a negative simple path which is what we need. However, we can do this latter part for shortest cycle (or detect negative cycles!) Shortest cycle (or detect negative cycles!) poly time solvable.

$\therefore$ Ratio Cycle problem is poly time solvable.

There is no such difficulty for Ratio trees, Ratio Perfect Matching etc.

This result is immensely useful in many contexts.