# CS6381: Lecture # 8 : 8.30.2020

## Synthesis of Network (Design)

Input : $r_{i,j} = r_{j,i}$ $\quad i = 1 \ldots n$, $j = 1 \ldots n$

Required minimum of max flow between i and j in an undirected network [One-at-a-time]; $C_{i,j}$ Cost/unit capacity on edge i,j

Output: Undirected edge capacities $U_{i,j}$ ; $i = 1 \ldots n$, $j = 1 \ldots n$ (i.e. $U_{j,i} = U_{i,j}$ $\forall i,j$)

So that we can send $r_{ij}$ flow from i to j in such a graph $\forall i, j$ and

$$\text{Min} \sum_{j} \sum_{i<j} C_{ij} U_{ij}$$

## Linear Programming Formulation

$U_{i,j} \geq 0 \quad i < j \; ; \; i = 1 \ldots n, \; j = 1 \ldots n$ { Undirected graph capacity

$$\sum_{\substack{i \in X \\ j \in \bar{X}}} U_{i,j} \geq \max_{\substack{i \in X \\ j \in \bar{X}}} r_{i,j} \quad \forall \; X \subseteq V$$

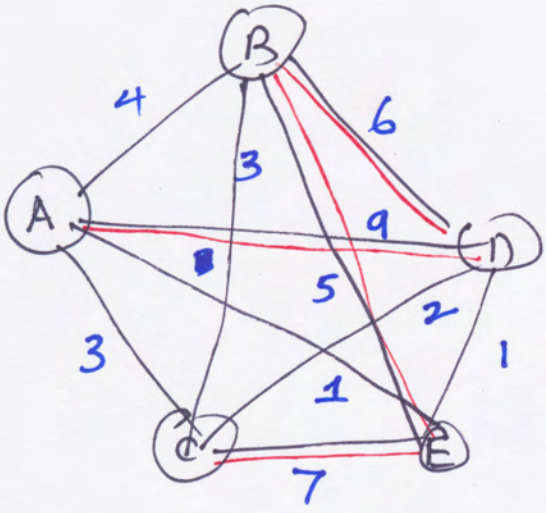
$$\text{Min} \sum_{i<j} C_{ij} U_{ij}$$

Even though the number of constraints is exponential, this problem can be solved in polynomial time by ellipsoid algorithm for example.

We consider here a special case which has an elegant solution due to T.C. Hu and R.E. Gomory [See also Mayeda, et al]. This algorithm of Hu-Gomory yields half-integral optimal solutions when the input data is integral. All this when $C_{ij} = 1 \ \forall \ i, j$. [Could be any constant but all must be equal]. There are two different algorithms ~~along the same lines~~ given by T.C. Hu & R.E. Gomory.

Let G be a complete graph on n vertices (normally denoted by $K_n$) with edge weights equal to $\{r_{i,j}\}$. This is called the requirements graph. (demand)

An example is shown on the next page



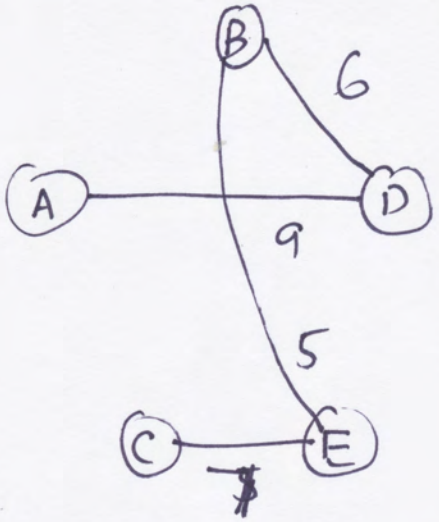

$r_{i,j}$ in blue

Requirements graph

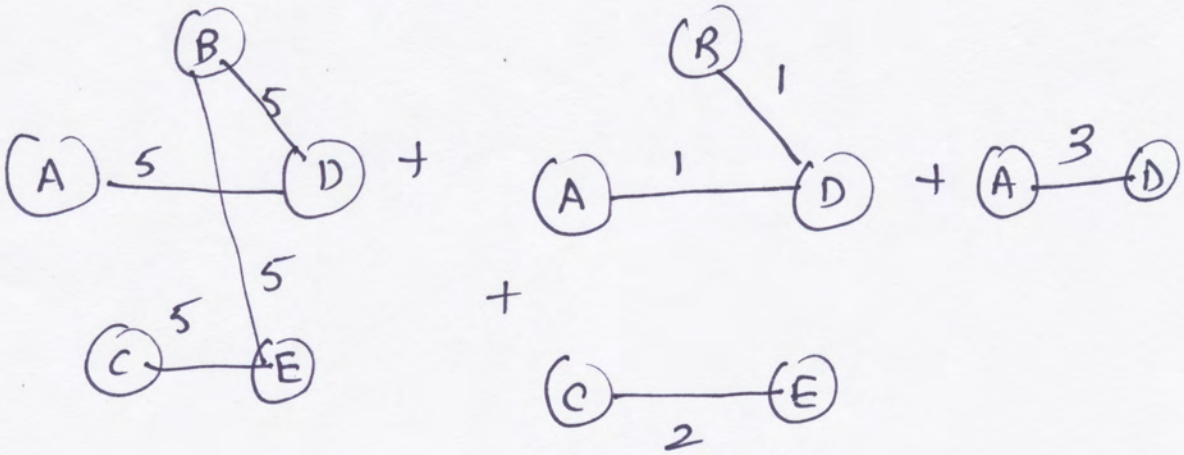
Max. Sp. Tree in red

Algorithm I

Step 1: Find a max. sp. tree in $(G, r)$

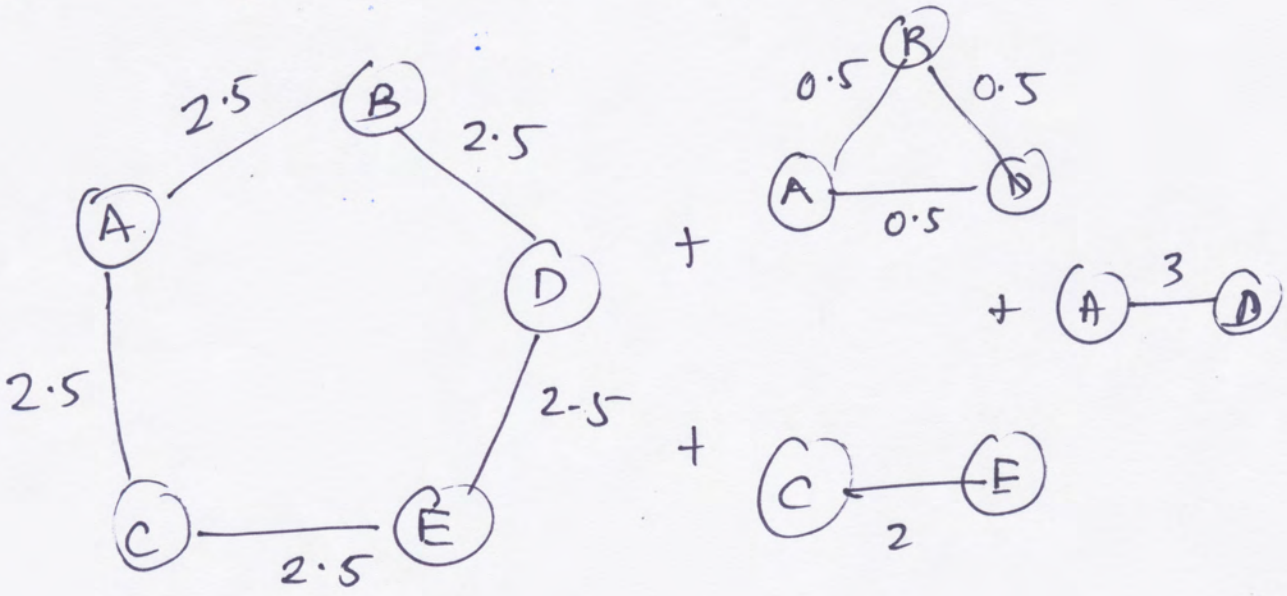



Step 2: Decompose into "uniform" trees

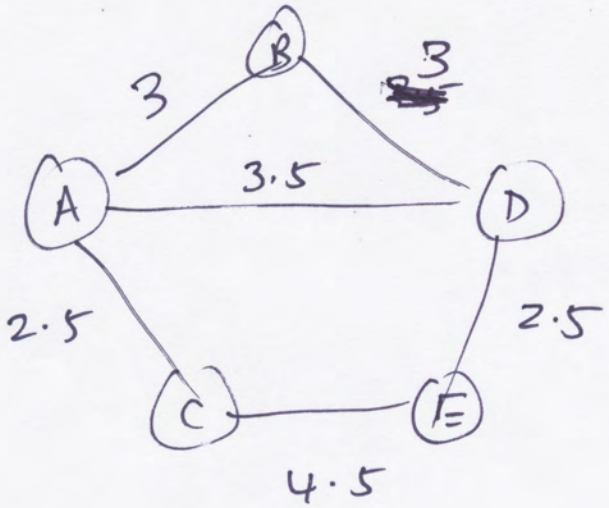


* Step 3 : For each Uniform tree, Construct a cycle on those nodes with half the value of edges in the tree

any? ↖



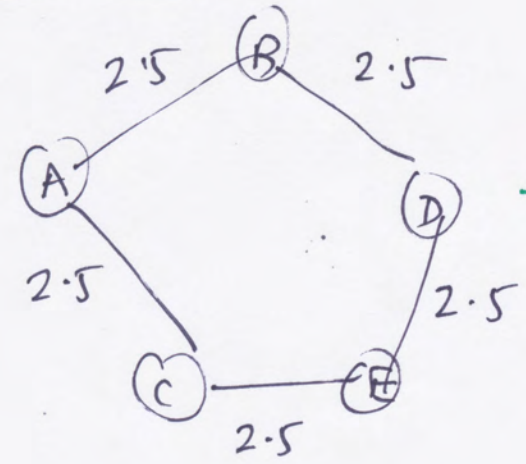

Step 4 Superpose all of these (i.e add capacities for each edge in each cycle) to get



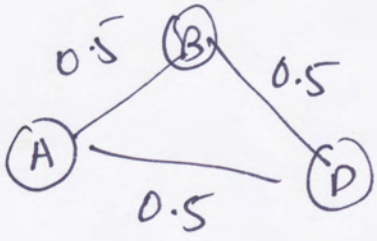

this is the output

## Proof of Connectness:

### Feasibility




→ provides for 5 units of flow between i and j $\forall$ i, j.




→ Provides for 1 unit of flow between i, and j : $i, j \in \{A, B, D\}$

∴ Overall network provides for the "dominant" requirements (that of max. sp. tree) and hence using triple inequality all demands can be met with this design.

Optimality needs a more elaborate proof which follows on next page.

LP has one constraint per set $X \subseteq V$.

$\therefore$ The number of constraints is $2^n$; $|V| = n$.

We can group these into two disjoint families

1) those with $|X| = 1$ ; i.e $X = \{i\}$ $i = 1 \ldots n$
(There are only $n$ of these)

2) All the rest.

If we discard all of (2), we get a "relaxed" problem; Any feasible solution to $\{(1) \& (2)\}$ is certainly feasible to $\{(1)\}$ but not conversely. $\therefore$ Set of feasible solutions to $\{(1) \& (2)\} \subseteq$ set of feasible solution to $\{(1)\}$ alone.

In general, an arbitrary optimal solution to $\{(1)\}$ alone may not be feasible to problem with $\{(1) \& \{2)\}$. However, Hu and Gomory found an optimal solution to problem with $\{(1)\}$ alone which happens to be feasible to problem with $\{(1) \& (2)\}$ and hence also optimal to this problem.

Revisiting LP with only n constraints:

$$u_{i,j} \geq 0 \qquad \begin{matrix} i = 1 \cdots n \\ j = 1 \cdots n \end{matrix} \; ; \; i < j$$

$$\sum_{j \neq i} u_{i,j} \geq \max_{j \neq i} r_{i,j} = \pi_i \qquad i = 1 \cdots n$$

$$\text{Min} \sum_{i<j} u_{i,j}$$

---

Example: $n = 4$.

$$u_{1,2} + u_{1,3} + u_{1,4} \geq \pi_1$$

$$u_{1,2} + u_{2,3} + u_{2,4} \geq \pi_2$$

$$u_{1,3} + u_{2,3} + u_{3,4} \geq \pi_3$$

$$u_{1,4} + u_{2,4} + u_{3,4} \geq \pi_4$$

Adding these we get

$$\sum_{i<j} u_{i,j} \geq \frac{1}{2} \sum_{i=1}^{n} \pi_i$$

← LB on the value of any optimal solution to problem with single node Constraints

Now we will show that our solution achieves this lower bound and hence is Min-Cost Solution.

For the graph constructed:

Max Sp. tree $T$ on the requirements graph is a set of dominant requirements — meaning if these are satisfied so are all others. This is because,

Since $T$ is a max Sp. tree,

$$v(x,z) \geq \min\left[v(x,y), v(y,u) \ldots, v(w,z)\right]$$

Where $x, y, u, \ldots w, z$ is the path in $T$ between $(x, z)$

$$\geq \min\left[r(x,y), r(y,u), \ldots r(w,z)\right]$$

Since dominant requirements are satisfied

$$\geq r(x,z)$$

Since $T$ is max-Sp. tree on $r$.

---

We have defined $\max_{j \neq i} r_{i,j} = \pi_i \quad i = 1 \ldots n$

If we look at $\max_{\substack{j: \\ (i,j) \in T}} r_{i,j} = \pi_i$

i.e at every node max edge must belong to the max. Sp. tree

The final graph that we produce, has the property that $\sum_{j \neq i} u_{ij} = \pi_i \quad \forall i$

$$\therefore \sum_{i<j} u_{i,j} = \frac{1}{2} \sum_{i} \pi_i$$

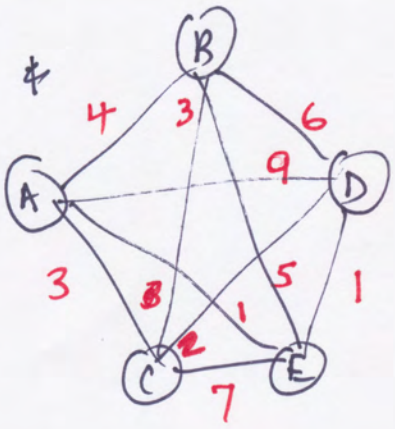
for our solution ; ∴ our solution is optimal for problem with constraints $\{1\}$ only.

But it is feasible for problem with $\{1\}$ and $\{2\}$.

Hence it is optimal for this as well.

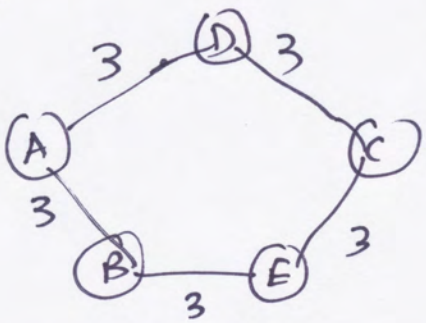
---

Algorithm II (We use the same example)




| $\pi_A$ | $\pi_D$ | $\pi_C$ | $\pi_E$ | $\pi_B$ (Sorted) |
|---|---|---|---|---|
| 9 | 9 | 7 | 7 | 6 |

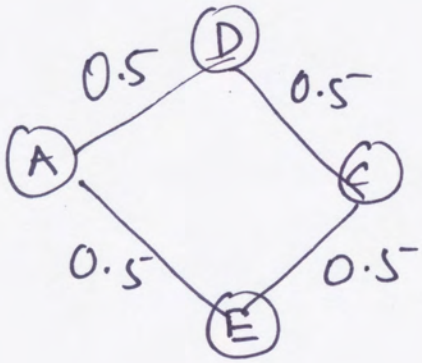
Step 1: For Cycle A - D - C - E - B ; $\frac{1}{2}(6) = 3$




This takes care of 6 units of flow for any pair.

| $\Pi_A^{new}$ | $\Pi_D^{new}$ | $\Pi_C^{new}$ | $\Pi_E^{new}$ | ~~$\Pi_B^{new}$~~ |
|---|---|---|---|---|
| 3 | 3 | 1 | 1 | ~~0~~ |

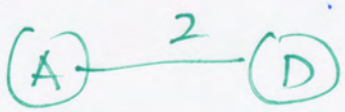
Step2: For cycle A-D-C-E : $\frac{1}{2}(1) = 0.5$




Adds 1 unit of flow for each pair in {A, D, C, E} bringing total to 7

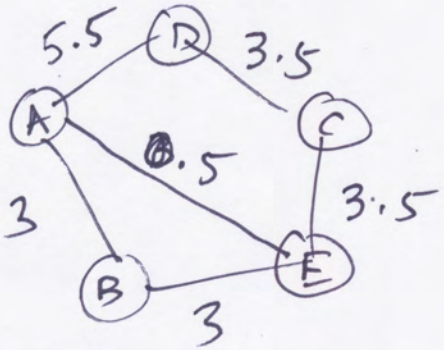
Step3 :

| $\Pi_A^{new}$ | $\Pi_D^{new}$ | ~~$\Pi_C^{new}$~~ | ~~$\Pi_E^{new}$~~ | ~~$\Pi_B^{new}$~~ |
|---|---|---|---|---|
| 2 | 2 | ~~0~~ | ~~0~~ | ~~0~~ |




Brings AD total to 9 as required.

Step4: Superpose to get




At each node

$$\sum_{j \neq i} u_{ij} = \Pi_i$$

∴ This is also optimal

This solution gives max flow without ↑ total cost.