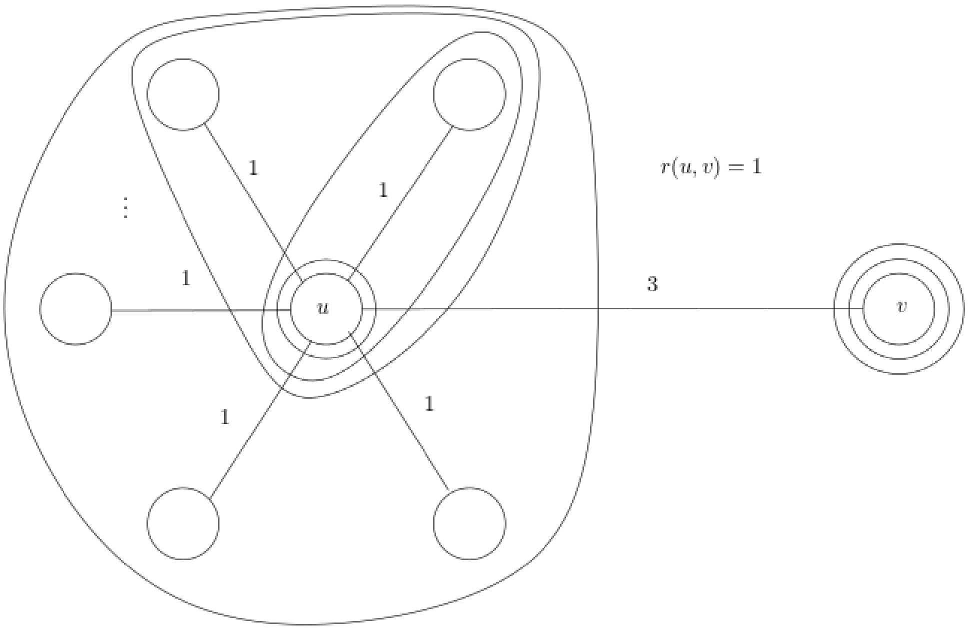
1 April 11th, 2019

Review

- At each iteration, the algorithm picks exactly one edge.
- The edges picked form a forest F.
- Active component are those that belong to $S^\star$.
- If F is not primal feasible, then there must be some connected component in F that is unsatisfied (active).
- We raise the dual simultaneously for all active components in a synchronized manner until some edge goes tight.
- We pick one of the edges and merge such components, terminating the iteration.
- We stop when a primal feasible solution F is formed (no active components).



Dual: $2 + 1 = 3$

Cost of solution: $3 + 1\times\#$ of spokes $= 3 + (|V| - 2 = |V| + 1 = \Omega(|V|)$

- As shown above, the solution might give us a terrible approximation ratio, which is why we have a pruning step to drop the redundant edges. After doing so, we would have cost of solution = 3.
- This can be done by seeing if $F - \{e\}$ satisfy the connectivity constraints, removing it if it is redundant. This will give us F'.

1.1 Steiner Forest Algorithm Correctness and Approx. Ratio

- **Correctness** - At termination, dual is feasible and primal is feasible.

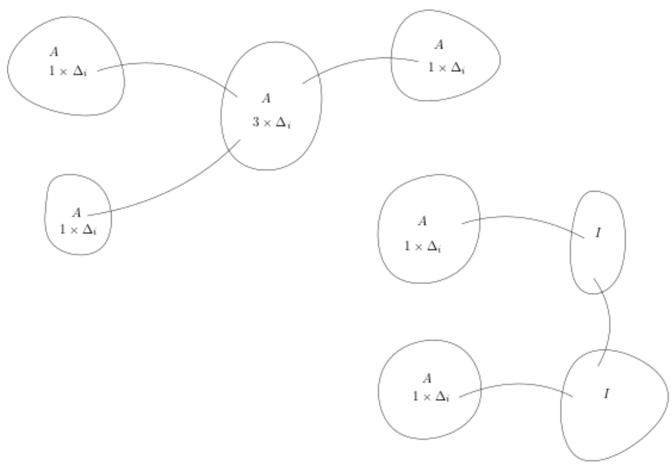
Proof.
- Dual is feasible because as soon as the dual constraint goes tight, the dual variable is frozen and thus the edges do not become over-tight.
- Primal is feasible because:
  1. Before the pruning step, the primal solution obtained is feasible (we stop only when all constraints of F are satisfied.
  2. The pruning step does not hurt feasibility. Note that for each pair of vertices u, v that have a connectivity constraint ($r(u, v = 1)$, there is a unique path between the two in F. Each edge e on that path will not be removed, as u, v would not be connected.

□

- **Approximation Ratio** The approximation ratio of this algorithm is 2.

Proof.
- By weak duality,
$$\sum_{S \in S^\star} y_s \leq OPT,$$
where OPT is the cost of the optimal Steiner Forest.
- Note:
$$\sum_{S \in S^\star} y_s = \sum_i (\Delta_i \times \# \text{ of active components}),$$
where Δ_i is the amount by which the duals of each active components is raised on the i-th iteration. (try to do this without using LP-duality)
- Now we'd like to show that $Cost(F') \leq 2 \times \sum_{S \in S^\star} y_S \leq 2 \times OPT$.
- Let us define:

$$\text{Degree of } S \subseteq V := \# \text{ of edges in } F' \text{ that belong to the cut } (S, \overline{S}).$$



– The above diagram demonstrates that:

$$Cost(F') = \sum_i \left(\Delta_i \times \sum_{S \text{ is active}} \text{Degree of } S \right).$$

– Suppose that Degree of $S \leq 2$, we would have our approximation ratio. (however this is not true, as shown above).

– Consider the average:

$$Cost(F') = \sum_i \left(\Delta_i \times \sum_{S \text{ is active}} \text{Degree of } S \right)$$

$$= \sum_i \left(\Delta_i \times \# \text{ of active components} \times \text{Average Degree of active components} \right).$$

meaning we have to show that the average degree of active components is less than or equal to 2.

– For a tree, it has n vertices and $n-1$ edges, meaning that average degree $= \frac{2(n-1)}{2} \leq 2$. So this is true for a forest.

– As such, if all components are active, then we are done, however this is not true in general, as an active component might have many inactive components connected.

– However, due to the pruning step, all inactive components must be internal.

– Consider the simple case first. Suppose there were no inactive components in iteration i. The average degree of the active components $\leq$ average degree of a tree ≤ 2.

– If there are inactive components that are leaves, we might be in trouble, as they would have degree 1, meaning that the average degree of active components could be ≥ 2.

– However, the clearly cannot be the case because of the pruning step.

– Thus any inactive component must have degree ≥ 2 as they are internal, meaning that the degree of active components ≤ 2.

Remark 1.1 — What if an inactive component has degree 0? (only consider the connected components with active component) TODO: think about this more

□

At each iteration, the algorithm picks exactly one edge.