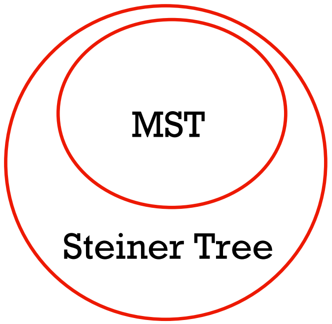
2-Approximation for Prize-Collecting Steiner Forest

Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, Mohammad Mahdavi

University of Maryland

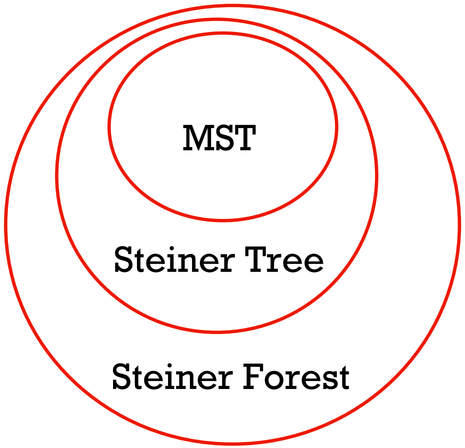
Steiner Tree

- Generalization of MST
- Weighted graph
- Set of vertices called terminals
- Connect terminals
    - Minimize total edge weight



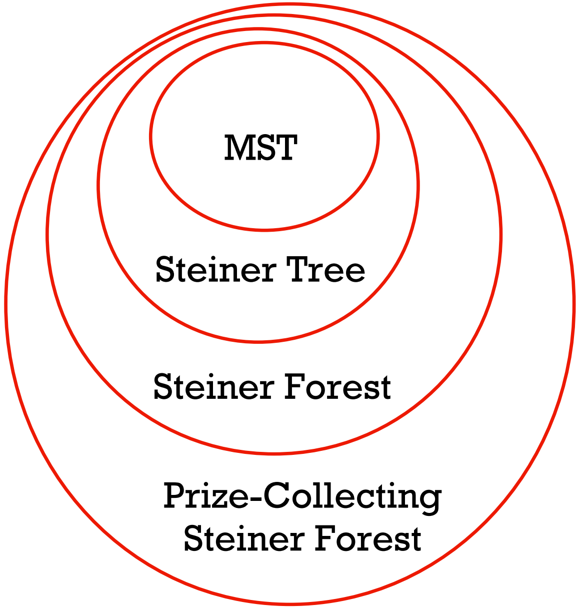
Steiner Forest

- Generalization of Steiner tree
- Set of pair of vertices called demand
- Connect vertices of each demand
    - Minimize total edge weight



Prize-Collecting Steiner Forest (PCSF)

- Generalization of Steiner Forest
- Set of pair of vertices called demand
- Each demand has a penalty
    - Either satisfy the demand
    - Or pay its penalty
- Minimize total edge weight + penalties paid



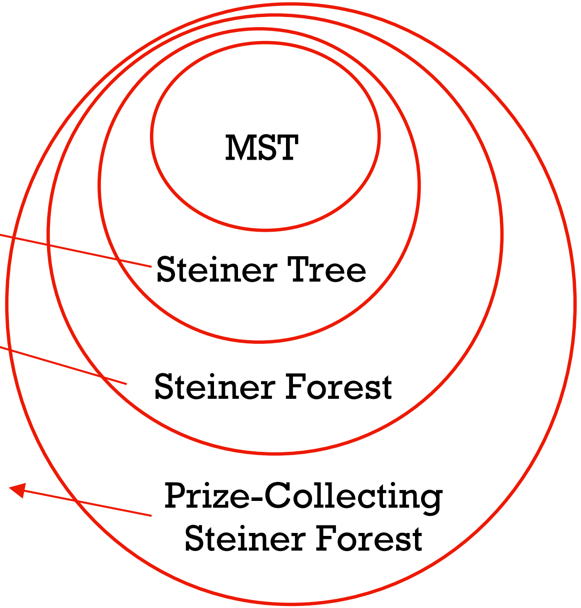
State of the Art

1.39-approximation [BGRS, STOC '10]

2-approximation [AKR, STOC '91][GW, SODA '92]

3-approximation via primal-dual [HJ, SODA '06]

2.54-approximation via randomize LP-rounding [HJ, SODA '06]



State of the Art

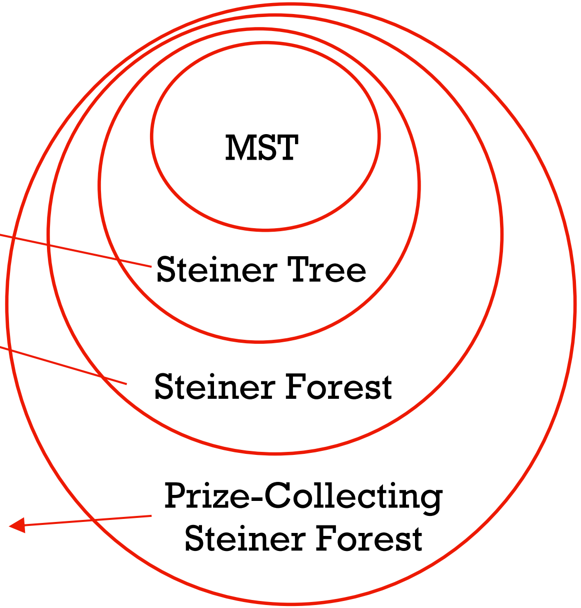
1.39-approximation [BGRS, STOC '10]

2-approximation [AKR, STOC '91][GW, SODA '92]

3-approximation via primal-dual [HJ, SODA '06]

2.54-approximation via randomize LP-rounding [HJ, SODA '06]

2-approximation via coloring schema



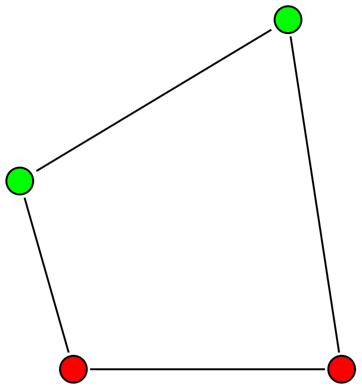
Our contribution

- Coloring schema
  - 2-approximation Steiner forest [GW, SODA '92]
  - 3-approximation PCSF [HJ, SODA '06]
- 2-approximation PCSF
  - Beat 2.25 integrality gap [KOPRSV, APPROX-RANDOM '17]

2-Approximation Steiner forest

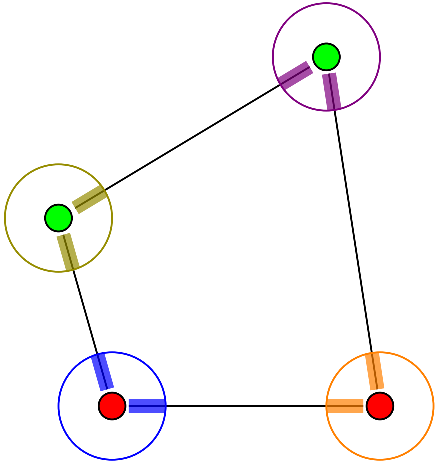
Definitions

- Edges are curve with length=weight
- We maintain a forest
  - Empty at the beginning
- Active sets
  - Connected components need to extend



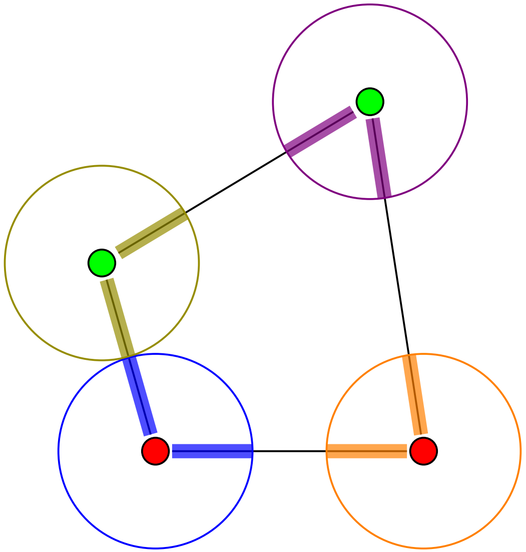
Static Coloring

- Each set has a unique color
- Active sets color their adjacent edges
  - Edges with exactly one endpoint in them



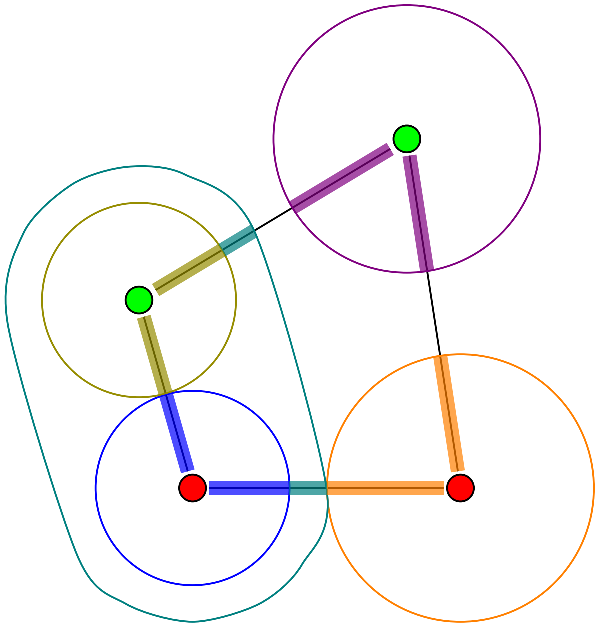
Static Coloring

- Each set has a unique color
- Active sets color their adjacent edges
  - Edges with exactly one endpoint in them
- One edge is fully colored
  - Add edge to our forest



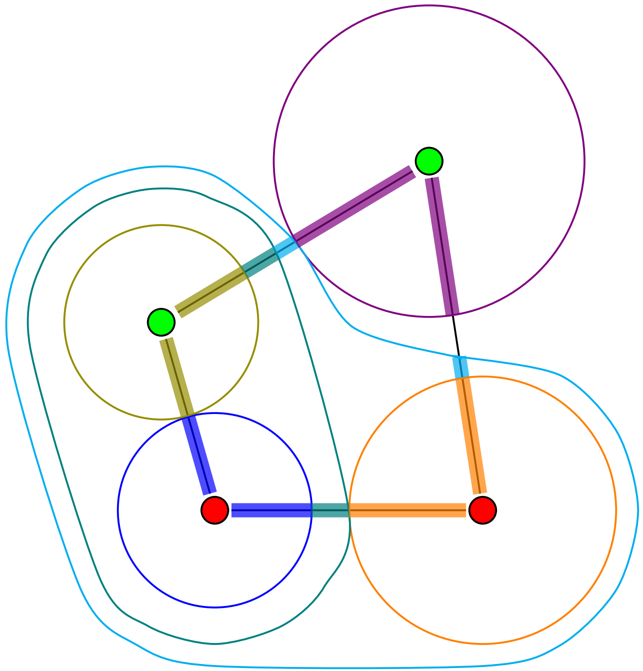
Static Coloring

- Each set has a unique color
- Active sets color their adjacent edges
  - Edges with exactly one endpoint in them
- One edge is fully colored
  - Add edge to our forest
  - Merges 2 connected components



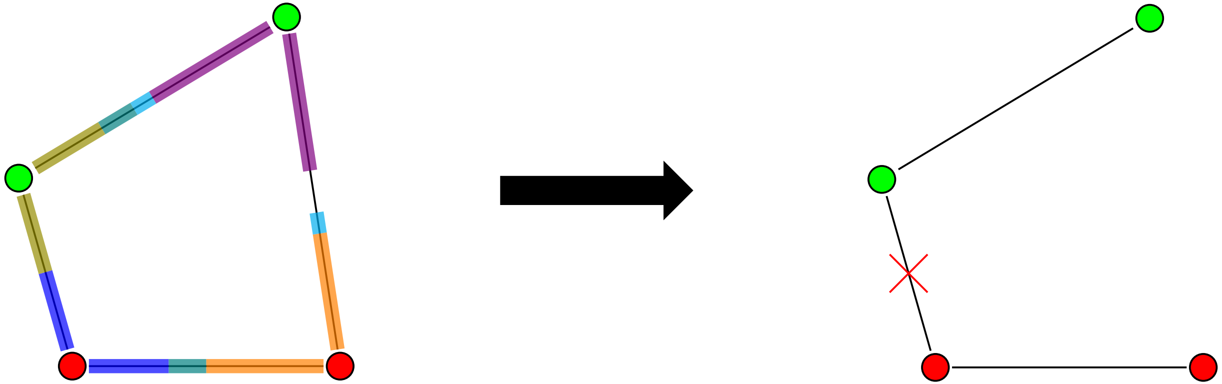
Static Coloring

- Each set has a unique color
- Active sets color their adjacent edges
  - Edges with exactly one endpoint in them
- One edge is fully colored
  - Add edge to our forest
  - Merges 2 connected components
- Finish when all demands are satisfied



Final Step

- Remove redundant edges



Analysis

- X: total coloring moment for all active sets
- Optimal solution is at least X
  - Active sets cut demands
  - They color every solution
- Our solution is at most 2X
  - At each moment, # edges are coloring ≤ 2 # active sets

3-Approximation Prize-Collecting Steiner Forest

PCSF

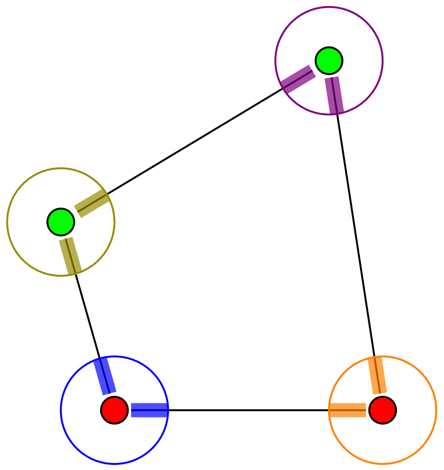
- Each pair has a penalty
- Don't connect a pair with a cost exceeding its penalty
- Dynamic coloring to maintain this constraint

Dynamic Coloring

- Each pair has a unique color
    - Limit colors by their pair penalty
- Run static coloring
- Assign each moment to a pair

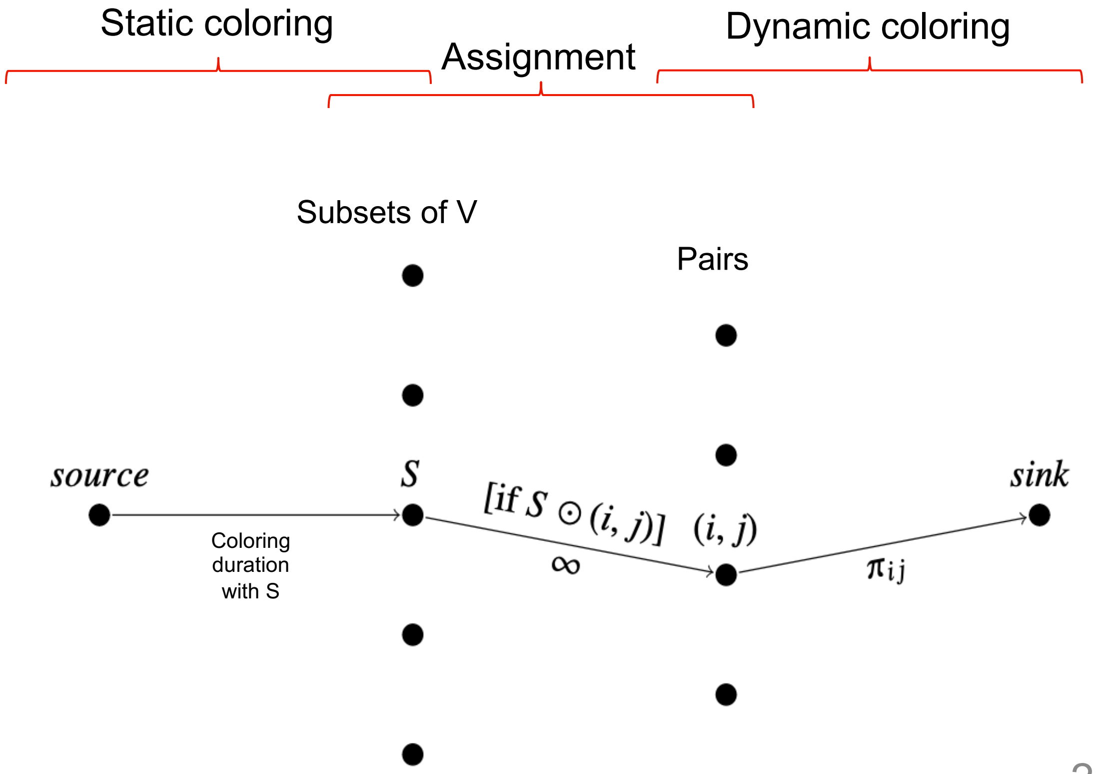
Dynamic Coloring

- Each pair has a unique color
  - Limit colors by their pair penalty
- Run static coloring
- Assign each moment to a pair
  - Active set cut the pair



SetPairGraph

- **Find max-flow**



Algorithm

- Run static coloring
- Hold if further assignment is impossible
  - Some pairs run out of their color (tight pairs)
    - Pay their penalty
  - Deactivate some active sets
- Repeat until there are no active sets

Final Step

- Consider max-flow with minimal tight pairs
- Pay penalty for tight pairs
- Connect other pairs
- Remove redundant edges

Analysis

- Our forest is at most 2 times the optimal solution
  - Similar to Steiner forest
- Our penalty is at most the optimal solution
  - Since the limit of using each color
- In total: 3-approximate solution

2-Approximation Prize-Collecting Steiner Forest

Iterative Algorithm

1. Run 3-approximation algorithm
2. If no penalties are paid, return
3. Remove demands if we paid them
   a. Pay their penalties in further solutions
4. Recursively run the algorithm on the new instance

Iterative Algorithm

1. Run 3-approximation algorithm
2. If no penalties are paid, return
3. Remove demands if we paid them
   a. Pay their penalties in further solutions
4. Recursively run the algorithm on the new instance

- Return the best solution

Intuition

- We paid tight pairs' penalty
  - Tight pairs color edges and their penalty are paid
- Analysis
  - Induction on the number of pairs with non-zero penalty
  - Comparing solution of recursive instance and input instance

Induction Base

1. Run 3-approximation algorithm
2. If no penalties are paid, return
3. Remove demands if we paid them
   a. Pay their penalties in further solutions
4. Recursively run the algorithm on the new instance

2-approximate solution
Similar to Steiner forest

- Return the best solution

Induction Hypothesis

1. Run 3-approximation algorithm
2. If no penalties are paid, return
3. Remove demands if we paid them
    a. Pay their penalties in further solutions
4. Recursively run the algorithm on the new instance

2-approximation for the recursive instance

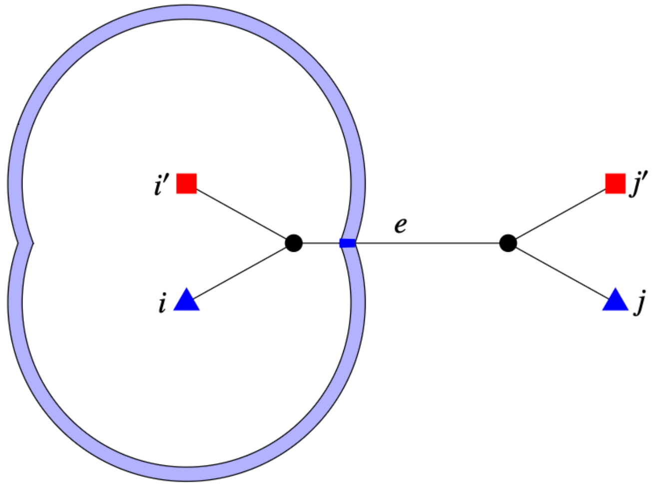
- Return the best solution

Induction Step

- Whether the cost of the optimal solution for the recursive instance is low
    - Recursive call gives 2-apprxomate solution
- Or the optimal solution has higher cost than our expectation
    - The 3-approximation algorithm has 2-approximate solution!

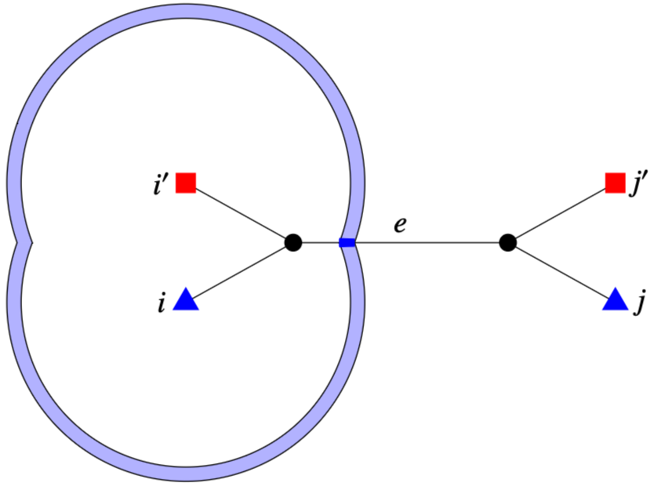
Review

- Tight pairs: pairs run out of color
- 3-approximation algorithm has minimal tight pairs
- Each assignment to a tight pair cannot be assigned to a non-tight pair



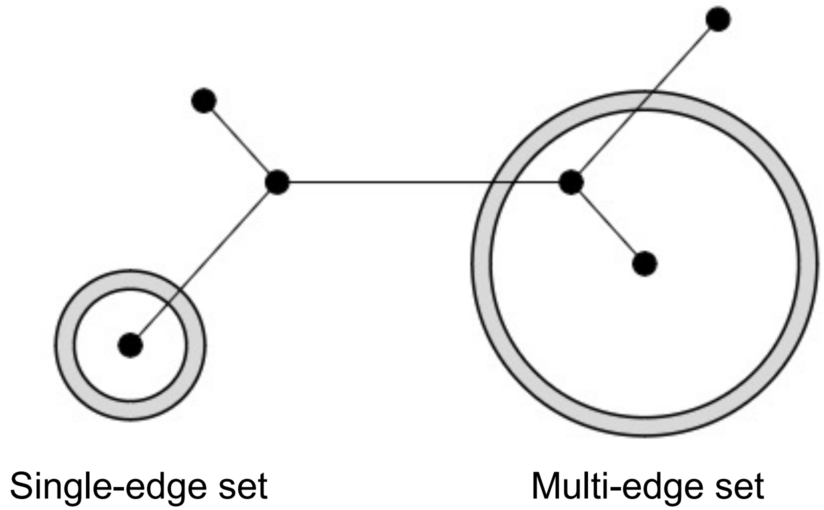
Removing Edges

- If an active set cuts only one edge of the optimal solution
- And its coloring assigned to a tight pair
- Remove the edge
  - Only disconnects pairs cut by the set
- Tight pairs removed from recursive instance
  - Still valid solution for recursive instance



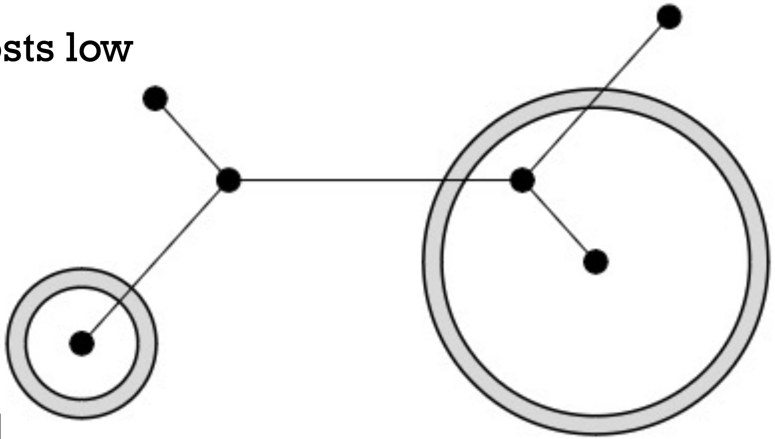
Divide Coloring with Tight Pairs

- How many edges in the optimal solution cut by an active set?



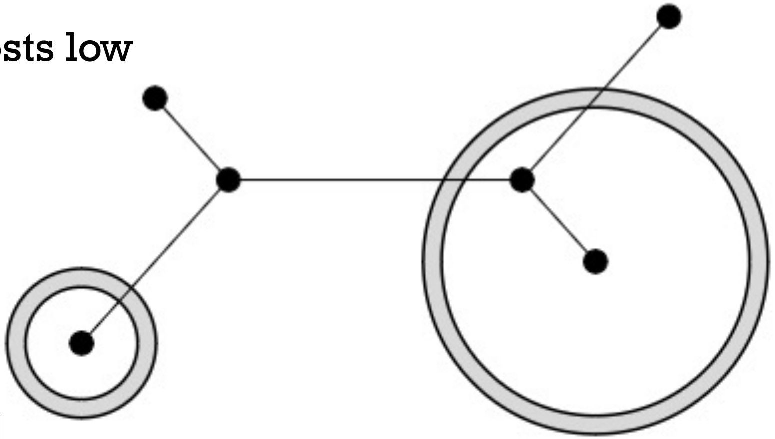
Finish Proof

- Singe-edge sets are majority
  - The optimal solution of recursive instance costs low
  - The recursive call costs low
- Multi-edge sets are majority
  - The optimal solution is large
  - The 3-approximation algorithm isn't that bad



Finish Proof

- Singe-edge sets are majority
  - The optimal solution of recursive instance costs low
  - The recursive call costs low
- Multi-edge sets are majority
  - The optimal solution is large
  - The 3-approximation algorithm isn't that bad
- Minimum of them is 2-approximate solution



Questions?

Thanks!

Registration and travel support for this presentation was provided by National Science Foundation.