Problema da árvore de Steiner

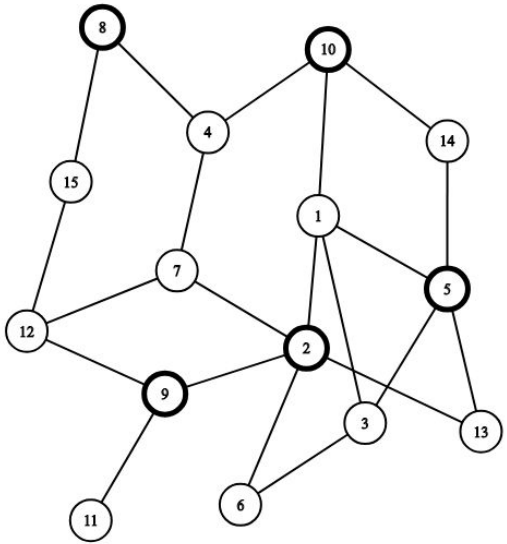
Variação: Restrição no número de roteadores e elos

Problema da árvore de Steiner

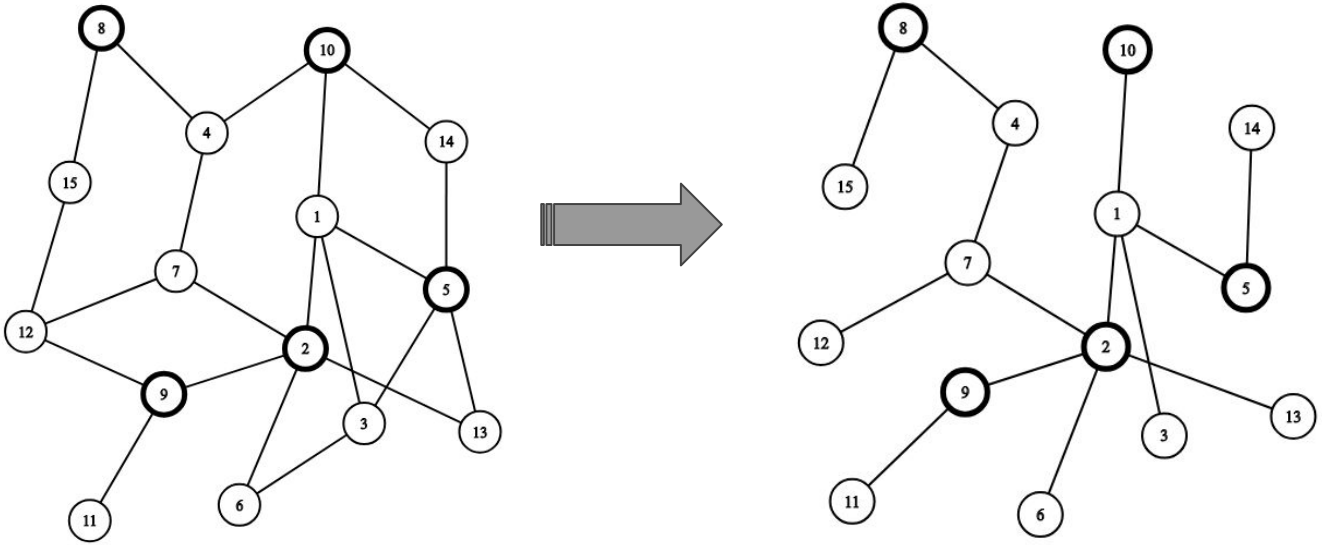
Heurística

- Construir Árvore
  - Método utilizado:
    - Algoritmo de Kruskal sem peso
    - Utilizando DSU para não criar ciclos
- Remover todas as folhas não terminais
  - Método:
    - Dfs a partir de um terminal(root) gerando as profundidades
    - Remover toda sub-árvore de um terminal que não contém terminais

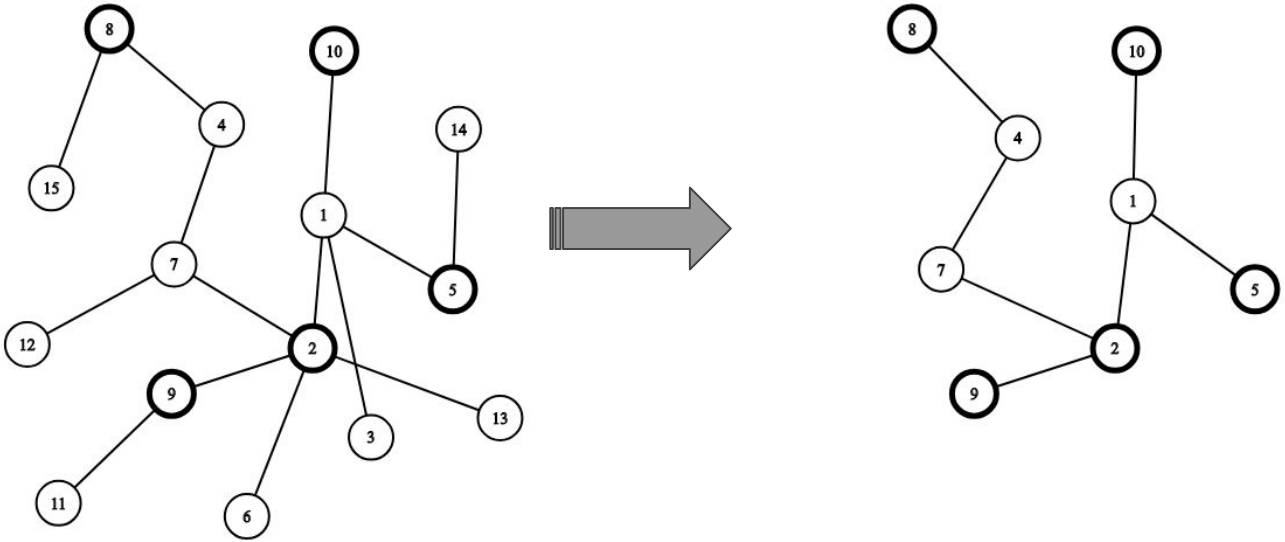
Grafo Inicial



Construção da Árvore



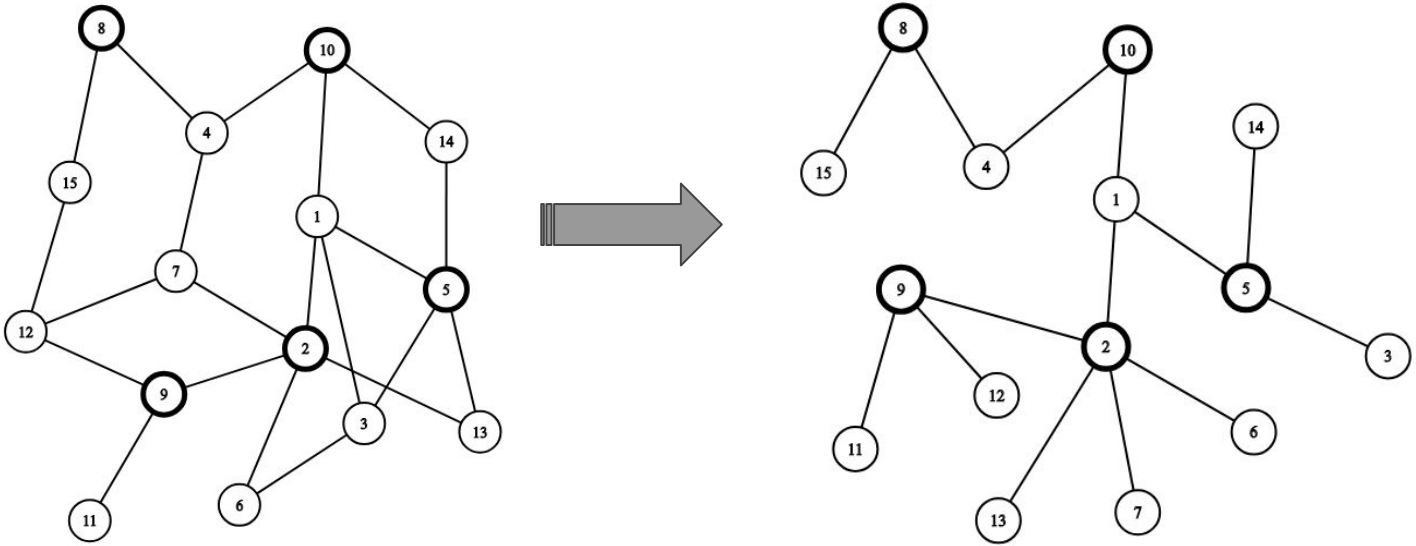
Remoção de folhas não terminais



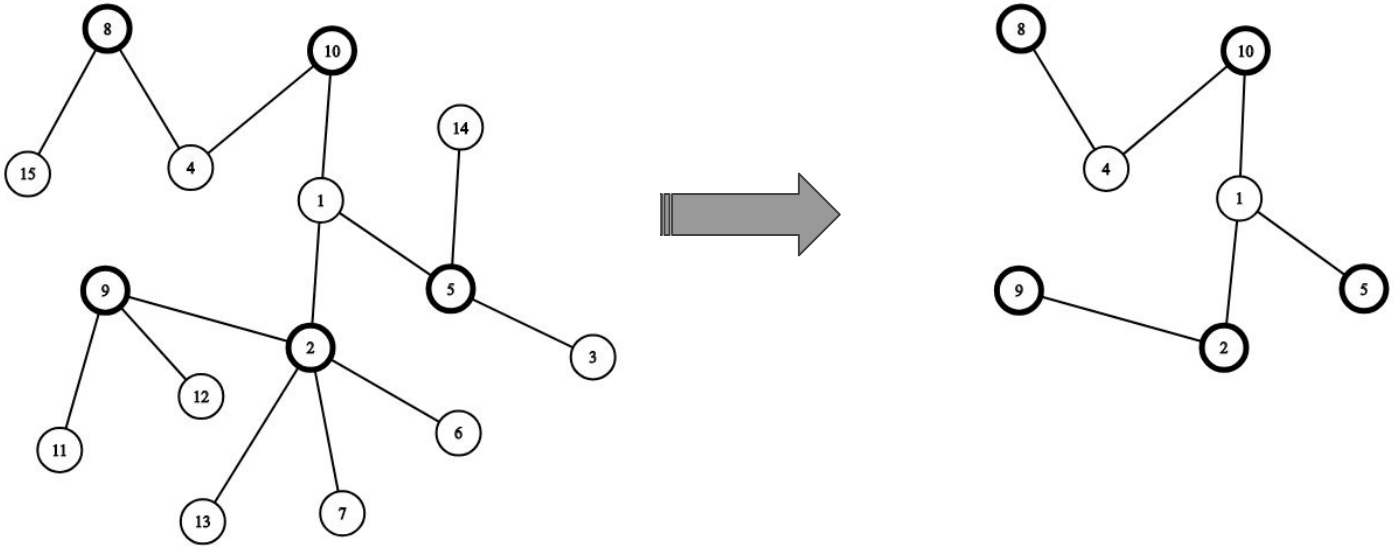
Melhoria na Heurística

Priorizar Arestas que contém nós terminais

Construção da Árvore



Remoção das folhas não terminais



Pré-Processamento

Pré-Processamento

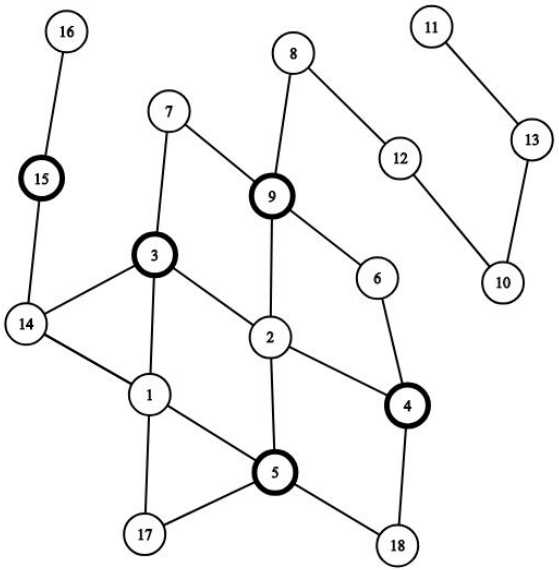
PRIORIZAR

1. Candidatos à roteadores : Nós ligados a muitos terminais
2. Candidatos à elos fortes: Nós que ligam dois terminais
3. Candidatos à elos fracos: Nós que ligam um terminal

DESPREZAR

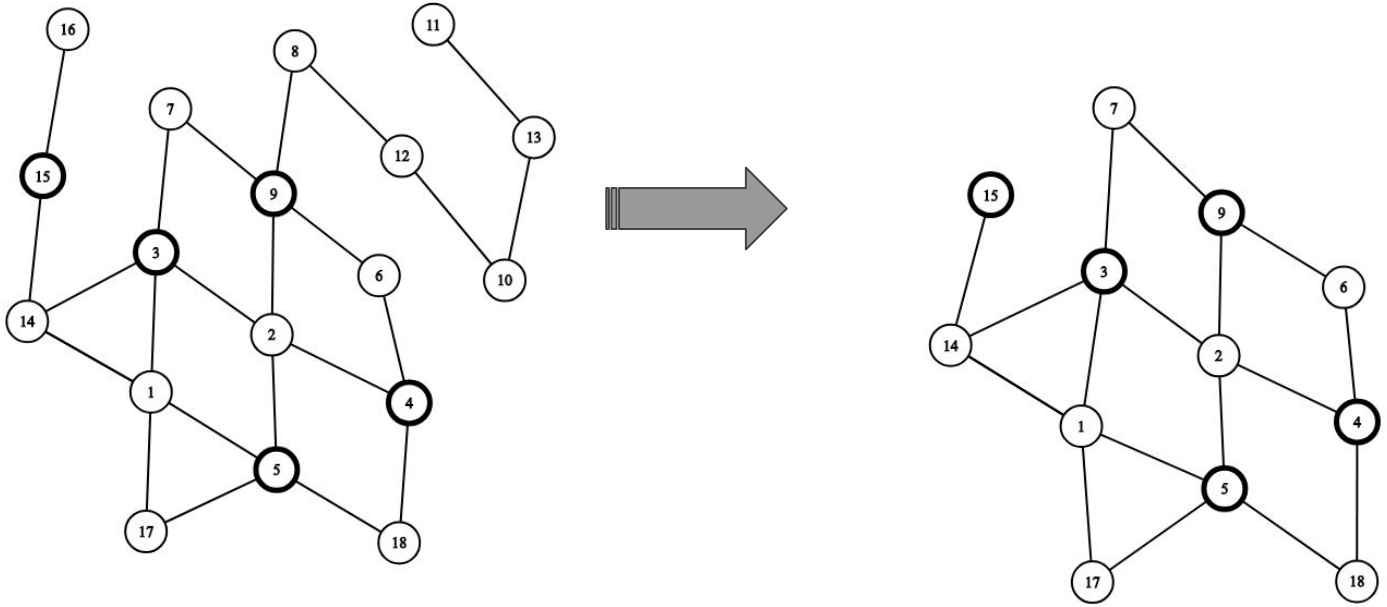
- Nós de grau 1 não terminais
- Nós de grau 2 em que seus vizinhos sejam ligados por uma aresta(essa parte requer uma forma eficiente de saber se essa aresta existe)

Pré-Processamento

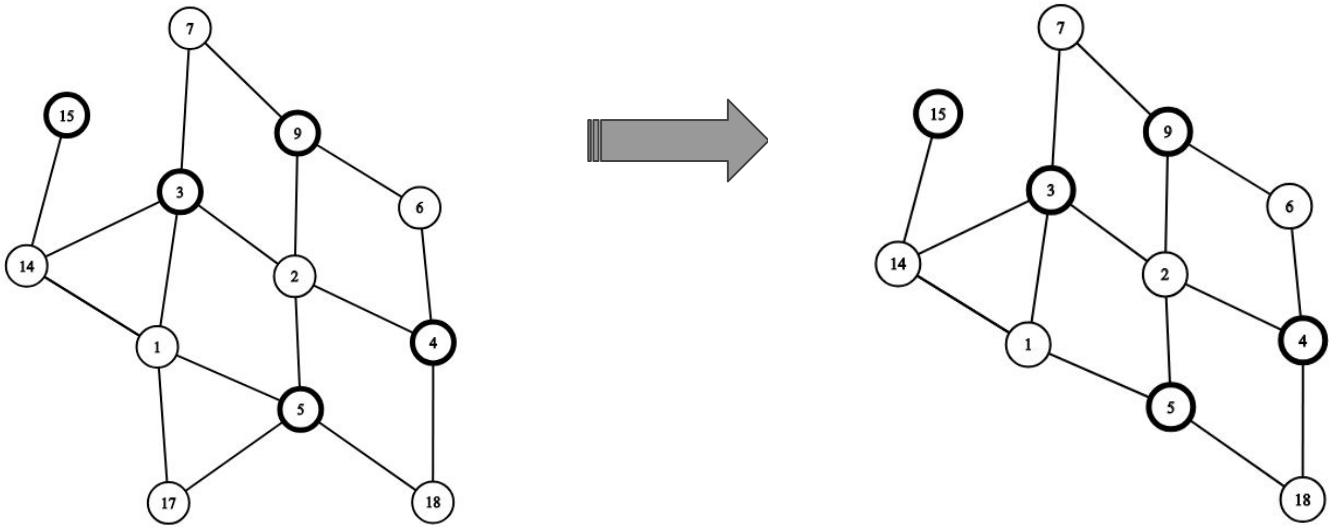


- Candidatos à roteadores
  - 2
- Candidatos à elos fortes
  - 14, 7, 1, 18, 6
- Candidatos à elos fracos
  - 16, 17, 8
- Removíveis pela regra do grau 1
  - 11, 13, 10, 12, 8, 16
- Removíveis pela regra do grau 2
  - 17

Pré-processamento: Regra do grau 1



Pré-Processamento: Regra do grau 2



Heurística

Uma abordagem usando GRASP

Heurística

- Agrupamento
  - Grupo 1: Candidatos a roteadores
  - Grupo 2: Candidatos a elos fortes
  - Grupo 3: Candidatos a elos fracos
  - Grupo 4:Todos os demais
- Grasp
  - Usando a ideia do kruskal construir uma árvore
    - Diferente do kruskal que usa o peso da aresta, usaremos aqui uma escolha pseudo-aleatório com peso, onde o grupo 1 tem prioridade sobre o 2 e assim por diante
  - Calcular penalidade
  - Busca Local

Árvore final