PAA **Nome:** Manassés
PPGCC - DCC - UFMG - 2015/01
Projeto Final
Prazo: 01/06/2015 **Professor:** Chaimo, Meira, Fabrício e Jussara

Etapas desse Trabalho

1. Descrição do Problema Prático de difícil resolução: Bottleneck Steiner Tree ($\mathcal{BST}$).
2. Modelagem do $\mathcal{BST}$ usando Teoria de Grafos. [2]
3. Implementação: uma solução exata (baseline) e um algoritmo aproximativo.
4. Análise de complexidade dos algoritmos.
5. Análise experimental dos algoritmos.

Descrição do Problema Prático

Bottleneck Steiner Tree $\mathcal{BST}$ Dado um conjunto $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ de n **Terminais** e um inteiro positivo k, obtenha a **Árvore de Steiner** contendo, no máximo, k **Pontos de Steiner** $\mathcal{S} = \{s_1, s_2, ..., s_k\}$ tal que o comprimento da aresta mais longa seja minimizado.[3]

Terminais Vértices localizados em uma dada posição no plano.

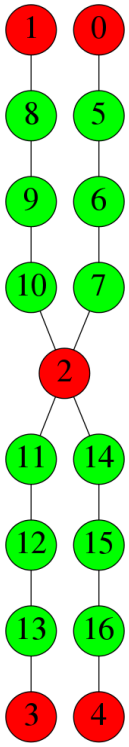
Árvore de Steiner Rede de comunicação acíclica entre os Terminais.

Pontos de Steiner Vértices *não-terminais* da Árvore de Steiner.

A **motivação** para estudar esse problema está na aplicação dele em projetos de *redes de comunicação sem fio*.

Suponha que, por restrições de orçamento, tenha-se que empregar apenas $n + k$ estações de transmissão. Sendo que n devem ser colocadas em locais pré-determinados. Nesse contexto, é interessante que a distância entre as estações seja a menor possível. O problema torna-se, portanto, escolher a melhor posição das k demais estações de modo a **reduzir a distância mais longa** entre elas.

Para ilustrar, considere os vértices ($\mathcal{S}$ e $\mathcal{T}$) e arestas da instância ao lado. Estabeleça peso positivo para as arestas, com valor exatamente igual a 1 entre $\mathcal{S}$ e $\mathcal{T}$ e no máximo 1 entre $\mathcal{T}$'s. Note que as arestas mais longas são aquelas que conectam os vértices terminais **7** com **11** e **10** com **14**, ambas com peso 2. Assim, uma solução com $k = 1$ seria selecionar o ponto de steiner **2** para compor a árvore de steiner com os demais terminais minimizando assim a aresta mais longa.



$\mathcal{S} = \{0, 1, 2, 3, 4\}$ e
$\mathcal{T} = \{5, 6, ..., 16\}$

Modelagem

O problema $\mathcal{BST}$ pode ser modelado da seguinte forma: dado um grafo $G = (V, E)$, um subconjunto $\mathcal{T} \subseteq V$ de terminais, uma função de distância (i.e. uma métrica) $d : E \to \mathbb{R}^+$ e um inteiro positivo k, encontre a árvore de steiner T com no máximo k vértices de steiner do subconjunto $\mathcal{S} \subseteq V \setminus \mathcal{T}$ tal que a maior aresta de T é minimizada.

Análise de Complexidade/Implementação

Algoritmo Exato

Encontrar na literatura um algoritmo exato (de implementação e compreensão acessíveis) para o $\mathcal{BST}$ demonstrou-se tarefa sem sucesso. Bem como conceber tal algoritmo. Assim, adotou-se a seguinte estratégia: construir a árvore de steiner minimal (problema para o qual foi encontrado algoritmo exato), observar quantos pontos de steiner foram necessários, usá-los como entrada para o algoritmo aproximativo, e comparar a maior aresta das duas soluções.

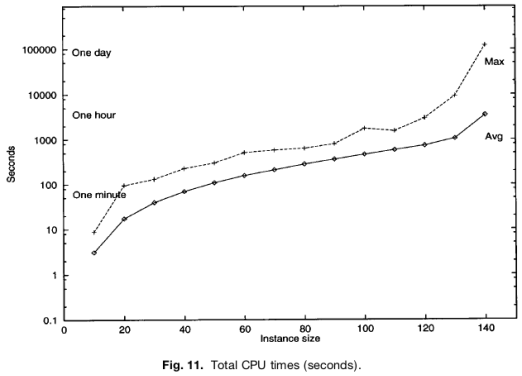


Fig. 11. Total CPU times (seconds).

Tempo de Execução [4]

Para tal será utilizado o trabalho sobre árvores de steiner minimais no espaço euclidiano [4] para o qual há o software GeoSteiner [1] de código acessível (por acessível entenda-se disponível para *download*, para compreendê-lo o degrau é inferior...).

Dizendo novamente, a saída do GeoSteiner, uma Steiner Minimal Tree (SMT) T^* terá a maior aresta comparada com a saída do algoritmo aproximativo proposto na seção seguinte, uma Steinerized Spanning Tree (SST) T para o mesmo número k de pontos de steiner.

Algorithm 1: steinerMinimalTree

Input: Grafo planar euclidiano $G(\mathcal{T}, E)$
Output: Aívore de Steiner Minimal T^* e um inteiro k

```
1 {T*, k} ← GeoSteiner(G)
2 return {T*, k}
```

GeoSteiner possui complexidade exponencial no o número de terminais n, trata-se de uma busca exaustiva que gera todas as possíveis árvores de steiner completas (FST) (fazendo podas quando possível) e realiza a concatenação delas de modo a produzir a árvore de steiner minimal (SMT) T^*. [4]

Algoritmo Aproximativo

O algoritmo aproximativo proposto utiliza-se da árvore geradora mínima para produzir uma solução para o $\mathcal{BST}$ que será no máximo duas vezes pior que o ótimo. O que significa produzir uma maior aresta até duas vezes maior do que a ótima.

Basicamente, explora-se uma estratégia gulosa, adicionando-se k pontos de steiner por iteração, àquela aresta cujo peso auxiliar l_e é o maior. O custo desse algoritmo é linear no número de arestas da árvore geradora mínima de G. Como essa possuirá $n-1$ arestas, têm-se portanto um algoritmo linear em n.

Algorithm 2: steinerizedSpanningTree

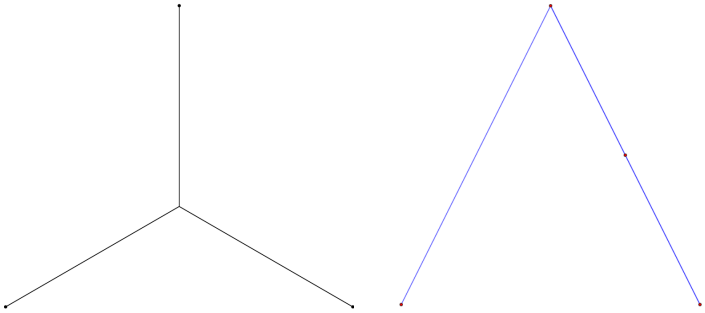
Input: Grafo planar euclidiano $G(\mathcal{T}, E)$ e um inteiro k
Output: Árvore geradora steinerizada T

$T(\mathcal{S}, \mathcal{E}) \leftarrow$ minimumSpanningTree(G) | $\mathcal{S} \leftarrow \mathcal{T}$, $\mathcal{E}_\mathcal{S} \subseteq E$, $n \leftarrow |\mathcal{T}|$
for $e \leftarrow 1$ **to** $|\mathcal{E}|$ **do**
  // $(v_i, v_j) \leftarrow e$ | $v_i, v_j \in \mathcal{S}$, $e \in \mathcal{E}_\mathcal{S}$, $i \neq j$, $i, j \in \mathbb{N}_{>0}$
  $\mathcal{P}_e \leftarrow \varnothing$
  $p_e \leftarrow 0$
  $l_e \leftarrow w_e$ // $w_e \rightarrow [0, \infty)$
for $s \leftarrow n+1$ **to** $n+k$ **do**
  $e \leftarrow \arg\max_{i \in [1, n-1]} \quad l_i$
  $\mathcal{P}_e \leftarrow \mathcal{P}_e \cup \{s\}$
  $p_e \leftarrow p_e + 1$
  $l_e \leftarrow \frac{w_e}{p_e + 1}$
for $e \leftarrow 1$ **to** $|\mathcal{E}|$ **do**
  if $p_e > 0$ **then**
    $(v_i, v_j) \leftarrow e$ // $i, j \in [1, n]$
    $\alpha \leftarrow$first$(\mathcal{P}_e)$
    $\mathcal{E} \leftarrow \mathcal{E} - \{e\} \cup \{(v_i, v_\alpha)\}$
    for ***each*** $s \in \mathcal{P}_e$ **do**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{v_s\}$ // $s \in [n+1, n+k]$
      if $\nu \leftarrow$next$(s, \mathcal{P}_e)$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v_s, v_\nu)\}$
    $\omega \leftarrow$last$(\mathcal{P}_e)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v_\omega, v_j)\}$
return T

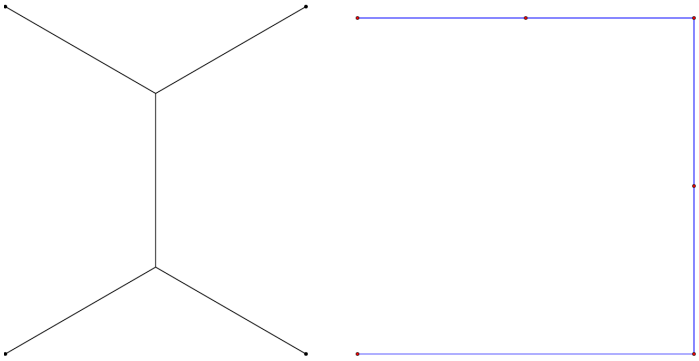
A implementação dos algoritmos foi feita na linguagem de programação python. Foram empregados script para execução de diferentes instanciações. Esses softwares estão disponíveis em https://github.com/mfer/sst.

Testes e Plano de Experimentos

Testes preliminares em instâncias de tamanho pegueno foram realizados com intuito de visualização do problema.

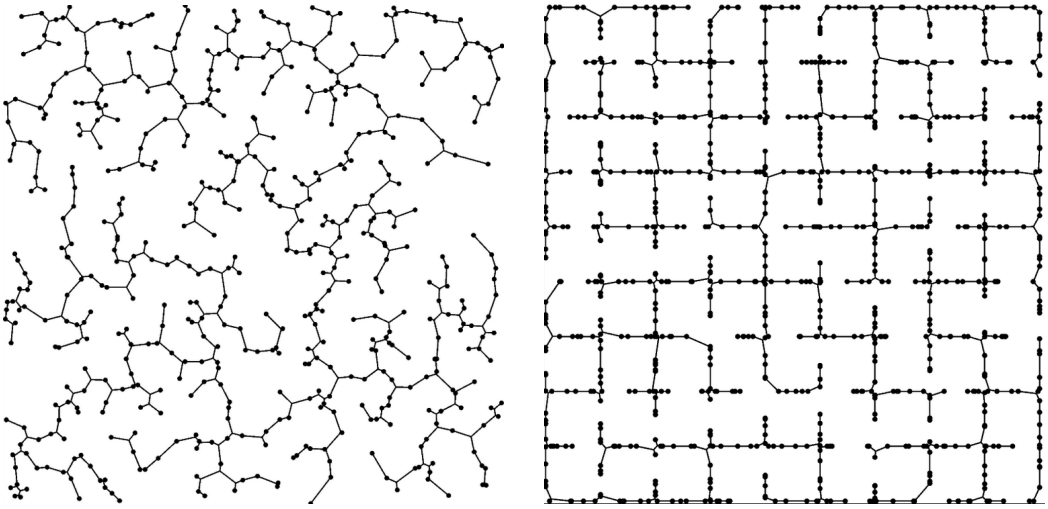


Steiner Minimal Tree (SMT) e Steinerized Spanning Tree (SST) para $n = 3$ e $k = 1$



Steiner Minimal Tree (SMT) e Steinerized Spanning Tree (SST) para $n = 4$ e $k = 2$

Serão gerados gráficos comparativos do baseline e do aproximativo para o tempo de execução e para a qualidade da solução $\frac{\max E(T^*)}{\max E(T)}$, tendo como parâmetro principal o tamanho da entrada em termos do número de terminais n.



Distribuição Aleatória e Regular numa Rede Quadrada

Serão realizados dois tipos de distribuições dos pontos terminais, a saber, uma aleatória e outra regular ao longo de uma rede quadrada. Deseja-se, principalmente, comparar o tempo de execução dessas realizações.

Análise Experimental

Os experimentos de simulação foram realizados com redes aleatórias. A geração dessas redes pode ser considerada etapa crítica para o experimento, no sentido de que ela tem custo alto. Para geração dos gricos abaixo em torno de cinco horas foram suficientes rodando no cluster do Laboratório Wisemap em uma máquina com 24 threads de processamento. Memória não é aspecto crítico. Foram paralelizadas as execuções de redes com parâmetros distintos.

Foram empregadas rotinas em R para tratamento analítico dos dados. Para cada boxplot apresentado foram realizadas 100 repetições daqueles mesmos parâmetros, basicamente três:

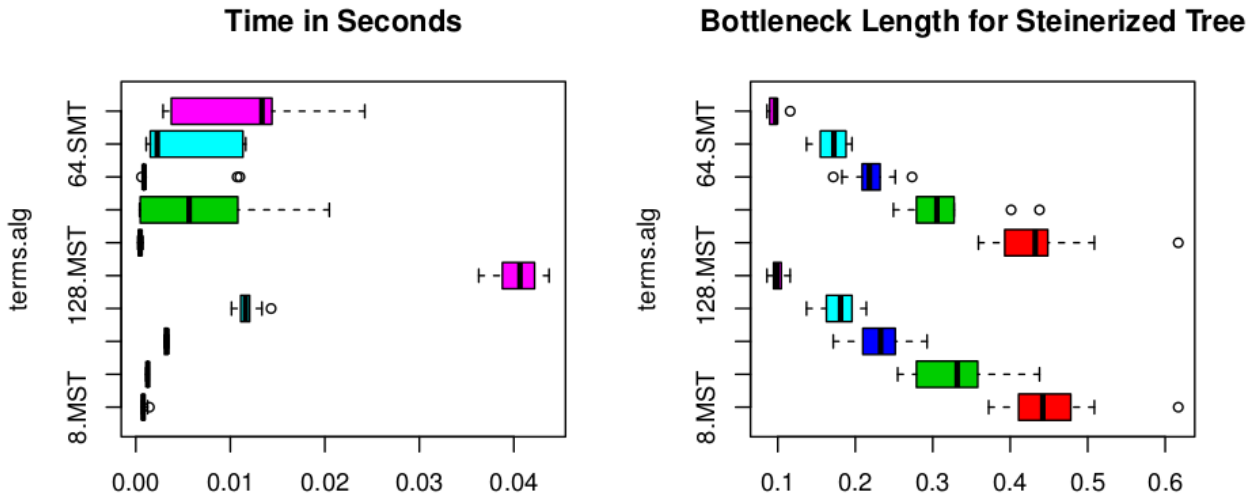
- **Algoritmo**:$\{SMT, MST\}$,
- **Distribuição**:{Regular,Aleatória} e
- **Quantidade de Pontos**:$\{8, 16, ..., 360\}$.

Quanto aos algoritmos, temos que SMT é a baseline. Consiste em obter a árvore de steiner mínima, isto é, rodar o algoritmo steinerMinimalTree e, em seguida, rodar o steinerizedSpanningTree para os k pontos de steiner e o grafo T^*. Assim emprega-se $2k$ pontos de steiner em uma árvore, k para torná-la mínima (comprovadamente a mais curta possível na soma total das arestas) e k para competir com o aproximativo na divisão das arestas.

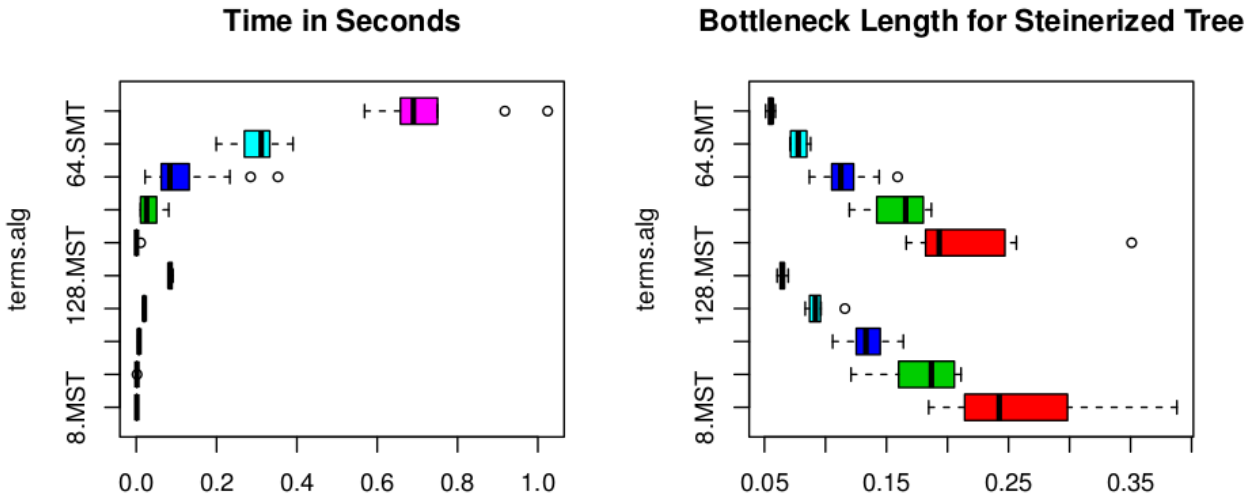
Isto é, o aproximativo MST irá diferir do baseline na árvore inicial que será uma geradora mínima (obtida por PRIM, por exemplo), nessa é empregado o steinerMinimalTree utilizando os mesmos K pontos retornados pelo steinerMinimalTree.

Distribuição e quantidade de pontos serão importantes para entender o comportamento dos algoritmos. Claramente SMT é exponencial enquanto MST é polinomial no número de pontos. No entanto, a distribuição regular torna o MST pior que o SMT em termos de tempo de execução.

Para os dois perfis de distribuição dos pontos foram obtidos resultados de tempo e qualidade em redes pequenas.

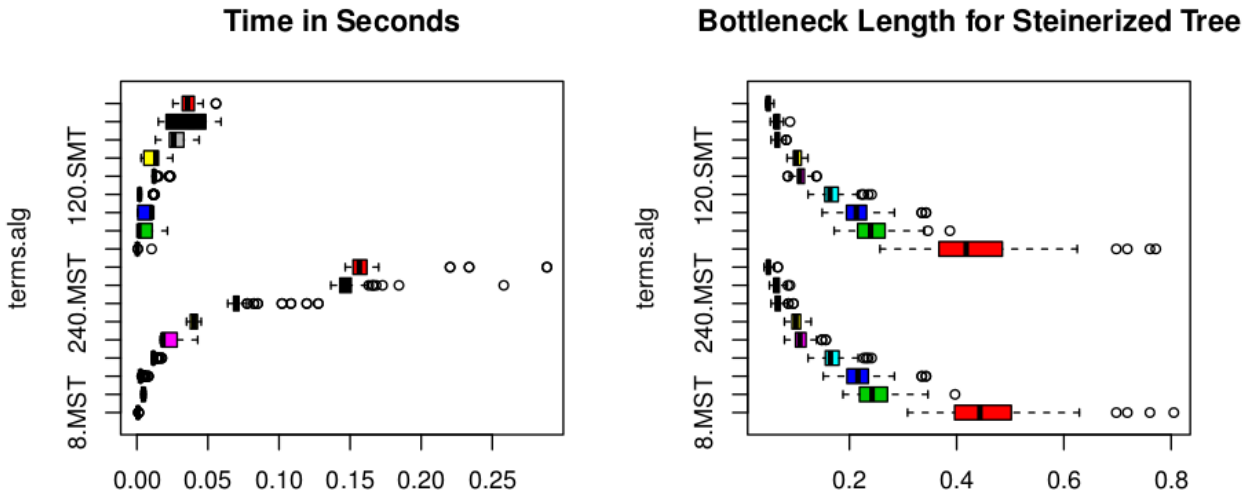


Tempo e Qualidade em grafos regulares



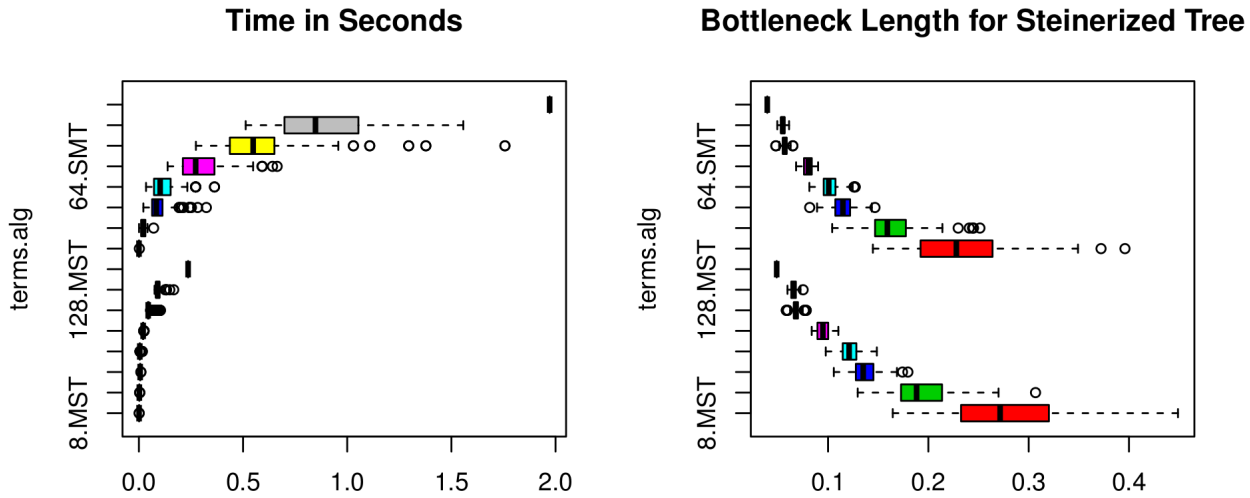
Tempo e Qualidade em grafos aleatórios geométricos

Para redes maiores, no caso regular, repete-se a desvantagem do MST perante o SMT em tempo de execução.



Tempo e Qualidade em grafos regulares

No caso aleatório, o tempo de execução do aproximativo é até oito vezes superior do que gasto pelo baseline. Já a qualidade dele é até uma vez e meia "pior". (Vale lembrar que o baseline não é o ótimo.)



Tempo e Qualidade em grafos aleatórios geométricos

Percebe-se que, para o caso regular, o aproximativo exibe tempos de execução maiores do que o baseline. Mas mantêm-se com qualidade 3/2 vezes inferior (isto é, o bottleneck é 3/2 vezes maior).

Conclusão

Uma árvore com menor maior aresta é aquela em que não existe uma segunda árvore geradora para qual a aresta de maior peso é menor do que a maior dessa primeira.

Essa maior aresta é conhecida como bottleneck (gargalo) da árvore.

Como a árvore geradora mínima e necessariamente uma árvore com menor maior aresta (isto pode ser provado por propriedade de cortes) os resultados para o algoritmo aproximativo MST são taõ bons. No entanto, uma árvore com menor maior aresta não sendo necessariamente uma árvore geradora mínima implica que o ótimo é melhor do que MST.

Para trabalhos futuros, pensa-se em propor ou encontrar uma melhor baseline para o $\mathcal{BST}$. Bem como, a geração de grande quantidade de números aleatórios é aspecto que deve ser melhorado.

Referências

[1] GeoSteiner software for computing steiner trees. `http://www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/`. Acessado: 2015-06-05.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[3] Wang and Z. Du. Approximations for a bottleneck steiner tree problem. *Algorithmica*, 32(4):554–561, 2002.

[4] Pawel Winter and Martin Zachariasen. Euclidean steiner minimum trees: An improved exact algorithm. *Networks*, 30(3):149–166, 1997.