Trabalho de análise e projeto de algoritmos: Árvore Geométrica de Steiner

André Igor Pereira[1], **Ricardo Buçard de Castro**[1]

[1]Escola de informática e computação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro – RJ – Brasil

Resumo. *A árvore geométrica de Steiner é um problema NP-Completo que é fonte de estudos matemáticos e computacionais há décadas, com problemas semelhantes tendo origem no século XVII. Baseado em um grafo no espaço R², busca-se uma árvore que percorra pontos terminais (pontos de Steiner) com a menor distância possível. Apresentaremos duas soluções uma gulosa e uma de força bruta para este problema.*

1. Introdução

O que se chama hoje problema da Árvore Mínima de Steiner é uma generalização de uma discussão antiga na matemática que já teve diversos nomes (Cavalieri, Fermat, Simpson, Torri- celli, Heinen, Viviani, Steiner, entre outros) que busca um ponto que minimiza a soma das distâncias para um conjunto de 3 pontos pelos quais precisa-se passar.

No mundo da computação, há diversos problemas a serem resolvidos através de algoritmos e otimizações desses algoritmos com o passar do tempo. Porém, há uma classe de problemas que ainda não foi possível de resolver de forma exata por limitações da computação da atualidade.

Neste trabalho, falaremos de desse tipo de problemas computacionais, a Árvore de Steiner, mais especificamente a variante Geométrica deste problema. Dividimos o trabalho nas seguintes seções: Introdução, Árvore de Steiner Geométrica, Propostas de solução e Considerações Finais. Na Introdução, falamos de conceitos importantes para entender o problema abordado, na seção Árvore de Steiner Geométrica, explicaremos a variante da Árvore de Steiner que adapta o problema para pontos em um espaço geométrico; já na Proposta de Solução, apresentamos dois algoritmos diferentes para solucionar o problema e na seção Considerações Finais, fazemos a conclusão do nosso trabalho.

1.1. Problemas NP-Completos

Antes de falar especificamente de problemas NP-Completos, é necessário explicar o que são problemas da classe P e problemas da classe NP. P são problemas que são resolvidos de forma determinística em um tempo polinomial, enquanto problemas da classe NP são para os quais são procuradas soluções aproximadas e eficientes de forma determinística[Goldreich 2010].

A teoria da NP-completude, é baseada na noção de uma redução eficiente de um problema para um outro para o qual seja possível criar um algoritmo eficiente para solucioná-lo. Portanto, um problema é NP-Completo se qualquer problema em NP é eficientemente reduzível para ele. Mostrar que um problema é NP-Completo implica que este problema não está em P a não ser que $NP = P$[Goldreich 2010].

1.2. Árvore de Steiner

O problema da Árvore de Steiner, também chamado de Árvore mínima de Steiner, é um problema relacionado à distância euclidiana. O problema é definido a partir de um conjunto N de de n pontos num plano euclidiano que, deseja-se achar o menor caminho possível entre pontos que interliga todos os n formando uma Árvore Mínima de Steiner (AMS)[Hwang and Richards 1992]. Além dos pontos pertencentes a N, a AMS também pode conter outros pontos, chamados de pontos de Steiner. Caso não sejam aceitos os pontos de Steiner na solução, seria uma Árvore Geradora Mínima (AGM)[Du et al. 2000].

Existem diversas variantes deste problema, como Árvore de Steiner Euclidiana, Árvore de Steiner Retilínea, mas, para apresentar um caso mais simples, falaremos da Árvore de Steiner em Grafos. Nesta variante, o objetivo é encontrar a AGM para um conjunto específico de vértices de um grafo chamados terminais[Hwang and Richards 1992].

Formalmente, dado um grafo ponderado $G = (V, E)$ e um conjunto $N \subseteq V$, a solução é um subgrafo de G com menor peso possível que gera N. Como falado anteriormente, vértices não pertencentes a N são chamados de vértices de Steiner caso tenham $grau \geq 3$ na Árvore de Steiner. Caso existam arestas negativas no grafo, há proposições de pré-processamento para para lidar com estas arestas para não favorecer percorrer estas arestas e diminuir o peso da Árvore de Steiner[Hwang and Richards 1992].

O problema da Árvore de Steiner foi provado ser NP-Completo no livro *Computers and Intractability: A Guide to the Theory of NP-Completeness* de 1979 de Michael R. Garey e David S. Johnson mostrando uma redução polinomial do problema 3-Satisfatibilidade (3-SAT) para o problema da Árvore de Steiner[Lewis 1983].

2. Árvore de Steiner Geométrica

O problema da Árvore de Steiner geométrica assume que são dados um conjunto de pontos em uma dimensão de tamanho n, e se deseja conectar alguns desses pontos pelo menor caminho possível. Os pontos que se deseja conectar são chamados de terminais, já os pontos que fazem parte do conjuntos, mas podem ou não ser usados para a confecção do caminho mínimo são chamados de pontos de Steiner.

O caminho formado pela ligação mínima desses pontos em um conjunto de dimensão n é uma árvore, daí o nome do problema árvore de Steiner. Ainda não foi encontrado um algoritmo para resolver o problema da árvore de Steiner de forma determínistica em tempo polinonial, logo esse problema, como será demonstrado adiante trata-se de um problema NP-difícil [Du et al. 2000].

3. Proposta de solução

Para resolver o problema de minimização contido na árvore de Steiner podem ser utilizados métodos não determinísticos, para que o problema possa ser tratado com uma complexidade polinomial Para o caso deste trabalho serão utilizadas técnicas, vistas durante o curso, que trazem complexidade exponencial. Serão propostas duas soluções uma usando um algoritmo de força bruta e a outra um algoritmo guloso.

3.1. Algoritmo Guloso

Um algoritmo guloso vai a cada passo em direção ao que parece melhor escolha para a resolução do problema em uma abordagem local, sem considerar o problema globalmente

[Cormen et al. 2012].

No problema da árvore de Steiner a escolha a ser feita a cada um dos pontos dados é qual caminho seguir. O algoritmo guloso escolherá sempre o passo com menor custo que o levará ao próximo ponto, podendo ser tanto um ponto de Steiner quanto um ponto terminal, mesmo que esse passo acabe não sendo o melhor se o problema for visto por uma avaliação global.

Para a abordagem utilizada o algoritmo recebe duas listas de pontos no R^2, com coordenadas x e y. A primeira das listas contém todos as pontos possíveis para um determinado problema, já a segunda traz quais os pontos da primeira são os terminais, um exemplo de instância de entrada pode ser visto na Figura 1.

```
pontos = [(0, 0), (2, 0), (1, 1), (1, 2)]
terminais = [(0, 0), (2, 0), (1, 2)]
```

Figura 1. Exemplo de instância de entrada.

O primeiro tratamento feito é transformar o conjunto de pontos em um grafo completo, ou seja, de cada vértice é possível acessar qualquer um outro. Para isso é feito um laço que itera sobre cada um dos pontos de entrada e cria uma aresta entre esse ponto e todos os demais. É atribuído um peso para cada aresta, esse peso é a distância euclidiana entre dois pontos no plano, definida como:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

Pode ser visto na Figura 2 um exemplo de grafo, gerado a partir da instância de exemplo, mostrada na Figura 1. No exemplo os terminais são os pontos pintados de vermelho, já os pontos de Steiner são mostrados em azul, e em cada aresta é exibida a distância euclidiana entre seus vértices adjacentes.

Com o problema transformado em problema de grafos é aplicado um algoritmo com abordagem gulosa, para encontrar a menor distância entre pares de pontos terminais. Primeiro é selecionado um ponto terminal como base, a partir desse ponto é são formados pares contendo esse ponto e mais um terminal, até que tenham se formado pares deste ponto com todos os outros terminais. A construção do algoritmo foi elaborada seguindo os seguintes passos:

Seguindo os passos a complexidade do algoritmo fica em (O($n^2 * m$)), sendo n o número total de pontos e m o número de terminais.

É possível perceber que a abordagem gulosa feita para este problema é capaz de achar um caminho que liga todos os terminais em uma árvore de Steiner geométrica, porém na maioria dos casos o resultado é uma árvore degenerada, que não apresenta ramificações, isso deve-se ao fato do algoritmo guloso estar sempre considerando o próximo passo como o vértice mais próximo, fazendo assim que muitas das vezes a escolha não seja a melhor escolha global. Uma das características também do algoritmo guloso é que depois de feita uma escolha ela não pode ser rejeitada a posteriori, por isso não foram tentados métodos para fazer uma segunda validação e melhorar o desempenho

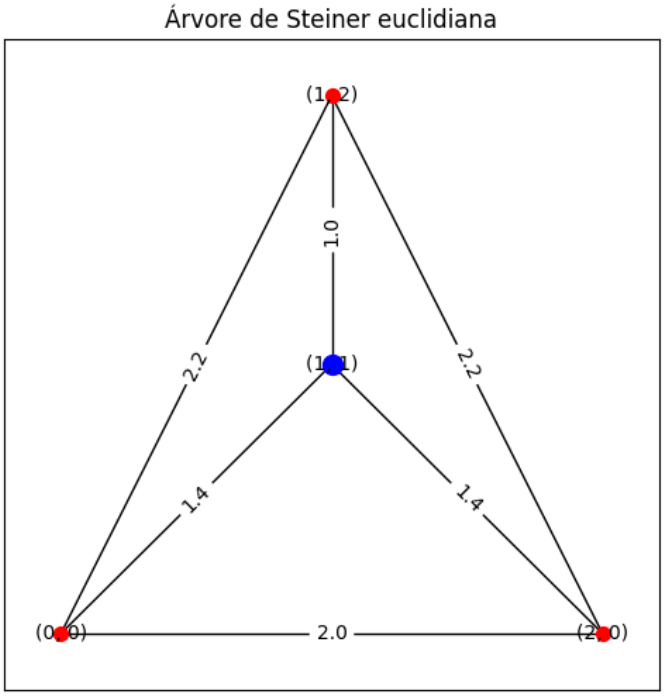

Figura 2. Exemplo de instância de entrada.

do algoritmo produzido. É mostrada na Figura 3 a árvore resultante, que é o caminho com as arestas em vermelho.

- Encontra na lista de pontos o primeiro terminal e o define como base
- Faz uma lista com as combinações entre o terminal base e todos os outros terminais
- A lista fica da seguinte forma:
  $[(base, terminal_1), (base, terminal_2), ..., (base, terminal_n)]$
- Para cada base e terminal encontra o menor caminho, usando abordagem gulosa:
  - Começa pela base
  - Vai para o vértice mais próximo
  - Adiciona a aresta percorrida nas arestas da arvore de Steiner
  - Confere se o vértice é o terminal de destino do par
    - Se for o destino, para a função
    - Se não for atualiza a base para o vórtice em que o programa está e refaz todas as etapas

3.2. Força Bruta

Algoritmos de força bruta partem de uma abordagem direta e exaustiva para solucionar um problema. Todas as possibilidades de solução de um problema são geradas e testadas, selecionando iterativamente a melhor solução a partir de um critério pré-determinado, sem nenhuma possível otimização de acordo com o domínio do problema.

Para a árvore geométrica de Steiner, temos um grafo ponderado gerado, sem laços, a partir de uma matriz de adjacência, em que cada vértice do grafo original recebe coordenadas x e y no R².

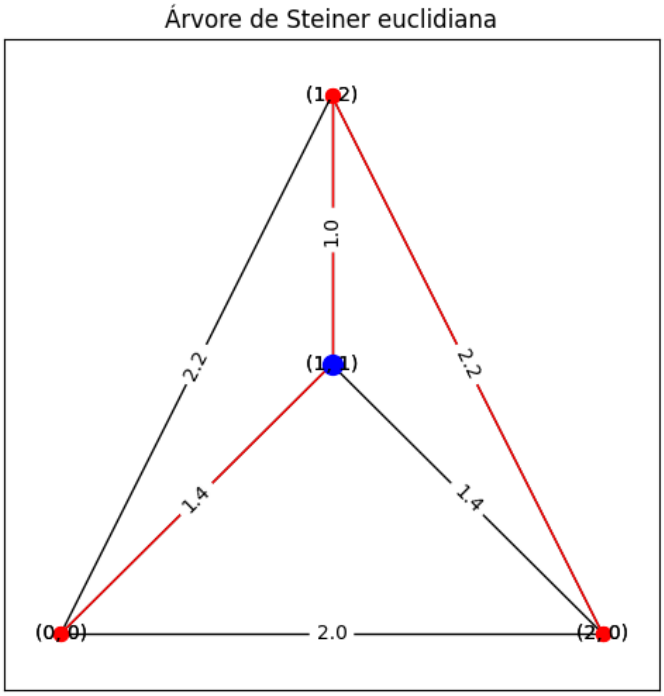

Figura 3. Árvore de Steiner.

O algoritmo então itera de 2 até o tamanho de terminais mais 1 vezes, criando subgrafos do grafo cheio. Após isso, são feitas combinações dos terminais com os subrafos criados. Baseado no ponto de origem e cada destino, é realizado um novo loop nos pontos do subgrafo para calcular o peso de cada aresta e mais uma iteração para adicionar cada aresta a uma árvore de Steiner. O custo total da árvore é calculado e então o algoritmo compara o menor custo atual de árvore com o custo da árvore gerada pelo loop: se for menor, a melhor árvore de Steiner recebe a árvore atual e o menor custo recebe o custo da árvore atual. Esse processo é repetido até acabar todas as combinações possíveis.

Com uma complexidade exponencial ($O(2^n)$) por conta do número de combinações realizadas com os terminais, o desempenho deste algoritmo é cada vez pior de acordo com o número de terminais, mas o número total de pontos também afeta o seu tempo de execução.

4. Considerações Finais

Neste trabalho foram construídas duas soluções usando abordagens diferentes para o problema da árvore de Steiner geométrica. Em nenhuma instância testada houve estouro de memória, para definir se o algoritmo roda em um tempo viável foi definido um limite de tempo de 3 segundos.

Na abordagem gulosa, como há o algoritmo tem uma complexidade menor foi possível rodar instâncias de no máximo 181 pontos totais e 121 pontos terminais. Instâncias testadas acima disso geraram um tempo médio de resolução maior que o tempo estabelecido. Na Figura 4 é possível ver um gráfico que relaciona o número de pontos, terminais e o tempo médio gasto para resolver a árvore.

Já utilizando a abordagem de força bruta a instância máxima que atendou os re-

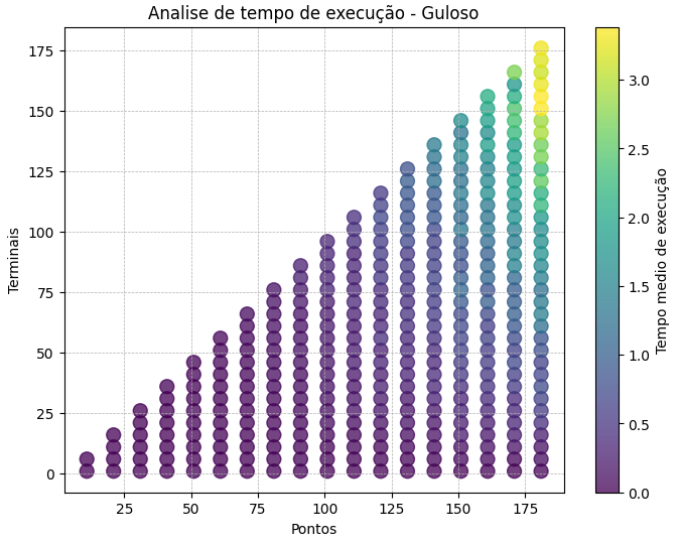

Figura 4. Gráfico de tempo de execução, abordagem gulosa.

quisitos de tempo tem 16 pontos com 10 terminais. A partir disso o tempo já dispara e passa do limite estabelecido. Esses números são exibidos na Figura 5.

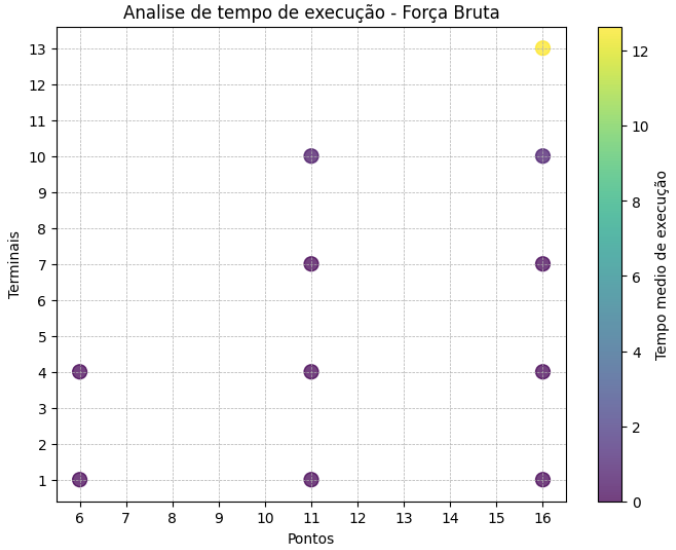

Figura 5. Gráfico de tempo de execução, abordagem força bruta.

Ambas as abordagens foram suficientes para resolver o problema, apesar da abordagem gulosa, devido às suas características, por vezes não conseguir a melhor solução, e a abordagem de força bruta tem dificuldade com instâncias maiores. Existem outras abordagens mais indicadas para esse problema, mas utilizam paradigmas não abordados nesse trabalho. Todo o material produzido pode ser encontrado no GitHub[1] do projeto.

[1]https://github.com/AnndreIgor/PPCIC/tree/main/Analise%20e%20Projeto%20de%20Algoritmos/Arvore%20de%20Steiner

Referências

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2012). Algoritmos-teoria e prática (3a. ediçao). *Editora Campus*.

Du, D.-Z., Smith, J., and Rubinstein, J. H. (2000). *Advances in Steiner trees*, volume 6. Springer Science & Business Media.

Goldreich, O. (2010). *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press.

Hwang, F. K. and Richards, D. S. (1992). Steiner tree problems. *Networks*, 22(1):55–89.

Lewis, H. R. (1983). Michael r. πgarey and david s. johnson. computers and intractability. a guide to the theory of np-completeness. wh freeman and company, san francisco1979, x+ 338 pp. *The Journal of Symbolic Logic*, 48(2):498–500.