**UNIVERSIDADE FEDERAL DE JUIZ DE FORA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**GABRIEL VIEIRA**

**CARLOS GUSTAVO FERREIRA**

**MARIA LUIZA DORNELAS**

**VITOR FERNANDES**

# TRABALHO 3 DE GRAFOS
# PROBLEMA DA ÁRVORE DE STEINER

JUIZ DE FORA- MG

2025

**SUMÁRIO**

## 1. DESCRIÇÃO DO PROBLEMA

Neste relatório, abordamos o Problema de Steiner, um dos desafios clássicos na teoria dos grafos e na ciência da computação. O problema consiste em encontrar a árvore de custo mínimo que conecta um conjunto específico de vértices, chamados de vértices terminais, em um grafo. Essa árvore pode incluir vértices adicionais que não são terminais (conhecidos como vértices de Steiner) para reduzir o custo total da conexão.

O Problema de Steiner é classificado como NP-difícil, o que significa que não se conhece um algoritmo eficiente (de tempo polinomial) para resolvê-lo em todos os casos. Por essa razão, a resolução exata do problema para grafos grandes ou com muitos vértices terminais torna-se inviável na prática, especialmente em aplicações que exigem respostas rápidas ou lidam com grandes volumes de dados.

Diante dessa complexidade, este relatório foca em abordagens heurísticas para encontrar soluções aproximadas. Essas heurísticas não garantem a solução ótima, mas são capazes de fornecer árvores de Steiner com custos reduzidos em um tempo computacional viável, mesmo para instâncias grandes do problema. As estratégias heurísticas exploradas incluem algoritmos gulosos, randomizados e reativos, que buscam equilibrar a qualidade da solução e o tempo de execução.

Além disso, o relatório discute a aplicação do Problema de Steiner em contextos práticos, como o projeto de redes de telecomunicações, o planejamento de circuitos integrados, a reconstrução de árvores filogenéticas em biologia computacional e a otimização de rotas em logística. Essas aplicações destacam a importância de encontrar soluções eficientes, mesmo que aproximadas, para problemas do mundo real que envolvem grandes quantidades de dados e restrições complexas.

Em resumo, o Problema de Steiner representa um desafio significativo na área de otimização combinatória, e este relatório explora estratégias heurísticas para lidar com sua complexidade, proporcionando soluções práticas e aplicáveis em diversos cenários.

## 2. DESCRIÇÃO DAS INSTÂNCIAS

Durante o processo de escolha das instâncias de rede, foi necessário realizar ajustes nas mesmas para garantir que estivessem em conformidade com o padrão de modelo exigido pelo algoritmo. As instâncias originais não estavam formatadas de maneira adequada para serem lidas de forma eficiente pelo algoritmo, o que exigiu a aplicação de transformações e ajustes nos dados. Esses ajustes foram essenciais para garantir que as informações fossem interpretadas corretamente e permitissem a execução de análises precisas e eficientes, respeitando os requisitos do modelo utilizado.

## 2.1) INSTÂNCIA 1:

- **Propósito:** A instância 1 pertence à categoria de redes biológicas e representa interações em organismos vivos, possivelmente modelando conexões neuronais, interações gênicas ou relações metabólicas na mosca-da-fruta. Essa rede pode ser usada para análises estruturais, como detecção de comunidades, centralidade de nós e identificação de padrões de conectividade, auxiliando pesquisas em biologia computacional e bioinformática.
- **Referência:** https://networkrepository.com/bio-grid-fruitfly.php
- **Descrição:**
  Grau: 176
  Ordem: 7282
  Direcionado: Não
  Componentes conexas: 61
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 49788

## 2.2) INSTÂNCIA 2:

- **Propósito:** A instância 2 representa uma rede biológica, onde os nós podem corresponder a entidades biológicas, como genes, proteínas, ou outras biomoléculas, enquanto as arestas indicam interações, relações funcionais ou estruturais entre essas entidades. O conjunto de dados é utilizado para analisar as interações complexas dentro de sistemas biológicos, como redes de expressão gênica, redes de proteínas ou até mesmo redes metabólicas. A topologia do grafo pode ser utilizada para estudar aspectos como a modularidade, a robustez do sistema biológico, e como as entidades se comunicam ou influenciam umas às outras.
- **Referência:** https://networkrepository.com/bio-dmela.php
- **Descrição:**
  Grau: 190
  Ordem: 7393
  Direcionado: Não
  Componentes conexas: 1
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 25569

### 2.3) INSTÂNCIA 3:

- **Propósito:** A instância 3 é um conjunto de dados de rede que pertence à categoria de Redes de Colaboração. Esse tipo de rede é comum em contextos acadêmicos e profissionais, onde os nós representam indivíduos, e as arestas indicam alguma forma de colaboração entre esses indivíduos, como coautoria de artigos ou participação conjunta em projetos. Essa rede pode ser usada para analisar a estrutura de colaboração dentro da comunidade científica, bem como para estudar a disseminação de ideias e inovações na área da física.
- **Referência:** https://networkrepository.com/ca-GrQc.php
- **Descrição:**
  Grau: 81
  Ordem: 5158
  Direcionado: Não
  Componentes conexas: 1
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 13422

### 2.4) INSTÂNCIA 4:

- **Propósito:** A instância 4 é um conjunto de dados de rede pertencente à categoria de Redes de Colaboração. Nessa rede, os nós representam pesquisadores e as arestas indicam colaborações entre eles, especificamente coautorias de artigos científicos. Essa rede pode ser usada para estudar a estrutura de colaboração no campo da matemática, bem como para calcular o "número de Erdős", que mede a proximidade de um matemático em relação a Erdős através de suas colaborações.
- **Referência:** https://networkrepository.com/ca-Erdos992.php
- **Descrição:**
  Grau: 61
  Ordem: 6100
  Direcionado: Não
  Componentes conexas: 1023
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 7515

## 2.5) INSTÂNCIA 5:

- **Propósito:** A instância 5 é um conjunto de dados de rede pertencente à categoria de Redes de Energia. Esta rede modela o sistema de distribuição de energia elétrica, representando nós como geradores, subestações ou pontos de consumo de energia, e arestas como as linhas de transmissão que conectam esses pontos. Essa rede pode ser usada para estudos de otimização de redes elétricas, análise de falhas, controle de fluxo de potência e melhoria na eficiência da distribuição de energia.
- **Referência:** https://networkrepository.com/power-bcspwr10.php
- **Descrição:**
  Grau: 14
  Ordem: 5300
  Direcionado: Não
  Componentes conexas: 1
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 13571

## 2.6) INSTÂNCIA 6:

- **Propósito:** A instância 6 pertence à categoria DIMACS10, que é um conjunto de dados utilizado em competições e estudos de otimização e análise de redes. No contexto de delaunay-n13, a rede é gerada com base em um grafo que segue a estrutura de uma triangulação de Delaunay, que é uma forma de dividir um espaço em regiões de maneira que nenhuma linha reta entre dois pontos da rede passe por dentro de um triângulo. Essa rede pode ser útil para resolver problemas de planejamento e otimização em grafos de forma eficiente, como minimização de distâncias, roteamento e problemas de conectividade.
- **Referência:** https://networkrepository.com/delaunay-n13.php
- **Descrição:**
  Grau: 12
  Ordem: 8192
  Direcionado: Não
  Componentes conexas: 1
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 24547

**2.7) INSTÂNCIA 7:**

- **Propósito:** A instância 7 pertence à categoria de Redes de Infraestrutura. Essa rede representa um sistema de distribuição de energia elétrica, onde os nós correspondem a elementos da infraestrutura, como geradores, subestações ou pontos de consumo de energia, e as arestas representam as linhas de transmissão ou conexões entre esses elementos. Esse tipo de rede é frequentemente usado em estudos de otimização, controle e análise de falhas em sistemas de energia, além de ser útil em modelagens de eficiência e distribuição de carga em redes elétricas.
- **Referência:** https://networkrepository.com/inf-power.php
- **Descrição:**
  Grau: 19
  Ordem: 5041
  Direcionado: Não
  Componentes conexas: 1
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 6594

**2.8) INSTÂNCIA 8:**

- **Propósito:** A instância 8 pertence à categoria de Redes Biológicas. Essa rede modela as interações entre proteínas de levedura (Saccharomyces cerevisiae), representando um exemplo de rede de interação proteica. Nessa rede, os nós representam proteínas, e as arestas indicam interações físicas ou funcionais entre elas. Esse tipo de rede é fundamental em estudos de biologia computacional, onde a análise de grandes redes biológicas pode revelar insights sobre doenças, desenvolvimento de terapias e funcionamento celular em organismos modelares.
- **Referência:** https://networkrepository.com/bio-grid-yeast.php
- **Descrição:**
  Grau: 2557
  Ordem: 6009
  Direcionado: Não
  Componentes conexas: 2
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 313890

### 2.9) INSTÂNCIA 9:

- **Propósito:** A instância 9 pertence à categoria de Redes do Facebook. Esta rede foi construída a partir de interações entre usuários na plataforma de mídia social Facebook e modela as amizades ou conexões entre um grupo de indivíduos. Os nós representam os usuários do Facebook, e as arestas representam as relações de amizade ou conexões sociais entre eles. Esse tipo de rede pode ser útil para entender como as interações sociais se espalham, identificar padrões de comportamento e estudar a formação e evolução das conexões sociais em plataformas de mídia social.
- **Referência:** https://networkrepository.com/socfb-Lehigh96.php
- **Descrição:**
  Grau: 973
  Ordem: 5075
  Direcionado: Não
  Componentes conexas: 2
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 198347

### 2.10) INSTÂNCIA 10:

- **Propósito:** A instância 10 pertence à categoria de Redes do Facebook. Essa rede foi criada com base nas interações sociais entre um grupo de 70 usuários do Facebook, modelando as amizades ou conexões entre eles. Nessa rede, os nós representam os indivíduos, e as arestas indicam as relações de amizade ou outras formas de conexão social. Esse tipo de rede é valioso em estudos de comportamento social, dinâmicas de comunicação e pode servir para entender melhor como as conexões sociais se formam e evoluem em plataformas como o Facebook.
- **Referência:** https://networkrepository.com/socfb-Vermont70.php
- **Descrição:**
  Grau: 864
  Ordem: 7324
  Direcionado: Não
  Componentes conexas: 2
  Vértices Ponderados: Sim
  Arestas Ponderadas: Sim
  Completo: Não
  Número de Arestas: 191221

## 3. DESCRIÇÃO DOS MÉTODOS IMPLEMENTADOS

- **Guloso Normal:**
  O algoritmo começa validando a lista de nós terminais, garantindo que estejam dentro do intervalo válido. Depois, ele inicializa algumas estruturas auxiliares, como arrays para armazenar predecessores, distâncias mínimas e um conjunto de nós que farão parte da árvore de Steiner.
  Em seguida, o algoritmo implementa manualmente uma fila de prioridade (min-heap), que será usada no Dijkstra para encontrar os caminhos mínimos entre os nós terminais. Essa estrutura permite inserir e remover nós de maneira eficiente, priorizando aqueles com menores distâncias.
  A principal etapa do algoritmo consiste em executar o Dijkstra para cada nó terminal. Ele percorre o grafo, calculando as menores distâncias até os outros nós e armazenando os predecessores para reconstruir os caminhos. Depois de executar o Dijkstra para um terminal, o algoritmo percorre os predecessores e adiciona as arestas encontradas à Árvore de Steiner.
  Durante esse processo, ele também marca os nós necessários para conectar os terminais. Isso significa que a Árvore de Steiner pode conter tanto nós terminais quanto outros nós intermediários, desde que eles ajudem a minimizar o custo total.
  Por fim, o algoritmo mantém um conjunto de arestas selecionadas, garantindo que nenhuma seja adicionada mais de uma vez. Ele também mantém o controle do peso total da árvore construída. Dessa forma, o resultado final é uma subárvore conectando todos os nós terminais com o menor custo possível dentro da estratégia utilizada.

- **Guloso Randomizado:**
  O algoritmo guloso randomizado introduz um elemento de aleatoriedade na escolha das arestas durante a construção da árvore. Em vez de sempre escolher a aresta de menor custo, ele seleciona aleatoriamente uma aresta entre as melhores opções disponíveis, com base em um parâmetro de controle chamado alpha. Esse parâmetro define o quão "gulosa" ou "randomizada" será a escolha. Um alpha próximo de 0 torna o algoritmo mais guloso (escolhendo sempre a melhor opção), enquanto um alpha próximo de 1 aumenta a aleatoriedade.
  No código, a função heapRandomTop é responsável por selecionar aleatoriamente um nó entre os melhores candidatos, com base no valor de alphaAtual. Essa abordagem permite explorar diferentes soluções, aumentando a diversidade das árvores geradas e, potencialmente, encontrando soluções melhores do que o algoritmo guloso tradicional.

- **Guloso Reativo:**
  O algoritmo guloso reativo é uma extensão do guloso randomizado que ajusta dinamicamente o valor de alpha ao longo da execução. Ele utiliza um conjunto de valores pré-definidos para alpha (no caso, {0.1, 0.2, 0.3, 0.4, 0.5}) e atribui probabilidades a cada um deles. Essas probabilidades são atualizadas com base no desempenho de cada valor de alpha em iterações anteriores. Valores de alpha que produzem soluções de menor custo têm suas probabilidades aumentadas, enquanto os que produzem soluções piores têm suas probabilidades reduzidas.
  No código, a função verifica se o algoritmo está sendo executado no modo reativo (reativo == true). Se estiver, ele escolhe um valor de alpha com base nas probabilidades atuais e atualiza essas probabilidades após cada iteração, com base no custo da solução encontrada. Essa adaptação dinâmica permite que o algoritmo se ajuste automaticamente para explorar os valores de alpha que tendem a produzir melhores resultados.

## 4. ANÁLISE DOS ALGORITMOS

Para a execução do código, cada instância foi executada 20 vezes, a fim de obter uma amostra representativa dos tempos de execução. Durante esses testes, foi medido o tempo de execução de cada instância para diferentes abordagens de algoritmo guloso, tanto para as representações de lista quanto de matriz. Para garantir a consistência dos resultados e minimizar a influência de variações aleatórias, foram usados 50 terminais gerados aleatoriamente, mas os mesmos terminais foram utilizados em ambas as representações, de lista e de matriz, para evitar que diferenças nos terminais influenciassem os resultados.

Além disso, foi calculada a média do tempo de execução para cada abordagem. Assim, esse processo permitiu comparar de forma mais precisa o desempenho das diferentes representações e identificar padrões relevantes na eficiência do algoritmo em cada caso.

Para evitar diferenças de desempenho causadas por variações de hardware e garantir um ambiente de execução padronizado, todos os experimentos foram realizados no GitHub Codespaces. Essa abordagem assegurou que os testes fossem executados em um ambiente de computação uniforme, reduzindo a influência de fatores externos, como especificações de máquina e carga de processamento, nos tempos de execução medidos.

| Tempo Médio de Execução(s) | | | | | | |
|---|---|---|---|---|---|---|
| Instâncias | Matriz | | | Lista | | |
| | Guloso | Randomizado | Reativo | Guloso | Randomizado | Reativo |
| 1 | 9.871760 | 24.093208 | 120.895680 | 24.489976 | 60.126029 | 292.561962 |
| 2 | 10.417211 | 25.598984 | 130.742928 | 29.625703 | 73.404119 | 350.446512 |
| 3 | 3.269873 | 9.963186 | 48.251234 | 6.693956 | 20.252085 | 102.607209 |
| 4 | 4.625637 | 11.267235 | 57.740229 | 7.414521 | 19.318351 | 89.791378 |
| 5 | 5.452823 | 18.958152 | 85.755088 | 7.117709 | 29.215313 | 127.503732 |
| 6* | 12.853904 | 110.498531 | 524.449390 | 25,212506 | 246.208285 | 1073.291579 |
| 7 | 4.710188 | 17.034061 | 79.022284 | 5.417667 | 19.157626 | 89.669919 |
| 8* | 7,672207 | 12,561312 | 62,063007 | 124.415094 | 194.955912 | 903.165699 |
| 9* | 5.756266 | 10.373101 | 49.703809 | 101.350196 | 166.629672 | 807.522416 |
| 10* | 11,114233 | 21,852123 | 109,721894 | 138,796378 | 251,610127 | 1221,281201 |

**Tabela 1 - Comparação entre o tempo médio de execução para cada instância e método.**

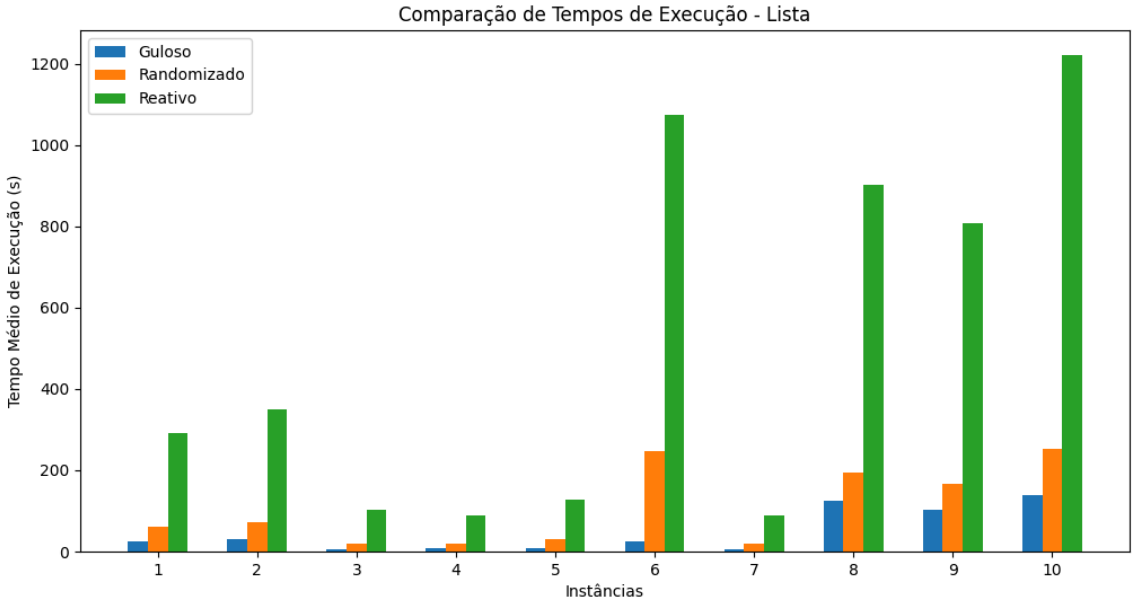


**Gráfico 1 - Gráfico referente à tabela 1 sobre a comparação entre o tempo médio para lista.**

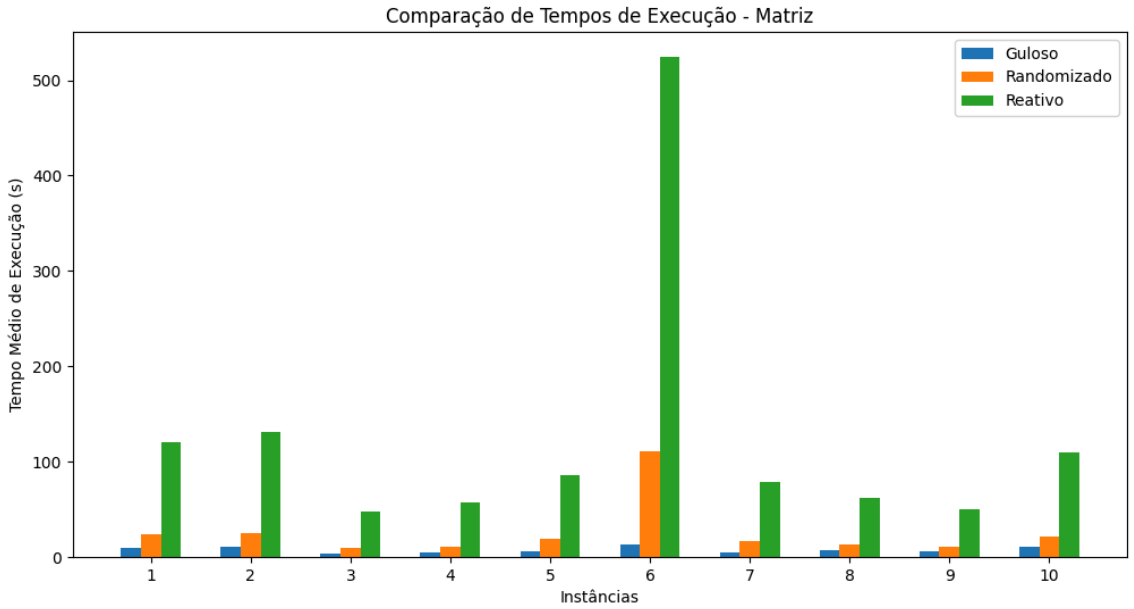


**Gráfico 2 - Gráfico referente à tabela 1 sobre a comparação entre o tempo médio para matriz.**

*Para as instâncias 6, 8, 9 e 10 o código foi executado apenas 3 vezes devido a maior densidade dos grafos.

## 5. ANÁLISE DE RESULTADO COM TESTE DE HIPÓTESE

- **Estrutura de Dados:**
  - Matriz: Representa a estrutura de dados usada para armazenar o grafo. Matrizes são geralmente mais eficientes para grafos densos, onde há muitas arestas.
  - Lista: Representa a estrutura de dados usada para armazenar o grafo. Listas de adjacência são mais eficientes para grafos esparsos, onde há poucas arestas.
- **Algoritmos:**
  - Guloso: Um algoritmo que faz escolhas locais ótimas em cada etapa com a esperança de encontrar uma solução global ótima.
  - Randomizado: Um algoritmo que introduz aleatoriedade nas escolhas, podendo melhorar a diversificação da busca e evitar mínimos locais.
  - Reativo: Um algoritmo que ajusta dinamicamente seus parâmetros (como a probabilidade de escolha de diferentes alphas) com base no desempenho das iterações anteriores.

- **Análise dos Tempos de Execução:**

- Comparação entre Matriz e Lista:
  - Em geral, os tempos de execução para a estrutura de Matriz são menores do que para a estrutura de Lista. Isso pode indicar que o grafo em questão é denso, onde a matriz é mais eficiente.
  - Por exemplo, na instância 1, o tempo de execução para o algoritmo Guloso com Matriz é 9,87 segundos, enquanto com Lista é 24,49 segundos.
- Comparação entre Algoritmos:
  - O algoritmo Guloso é o mais rápido em todas as instâncias, tanto para Matriz quanto para Lista. Isso é esperado, pois ele faz escolhas locais sem considerar o impacto global, o que reduz o tempo de execução.
  - O algoritmo Randomizado é mais lento que o Guloso, mas ainda mais rápido que o Reativo. A introdução de aleatoriedade aumenta o tempo de execução, mas pode melhorar a qualidade da solução.
  - O algoritmo Reativo é o mais lento em todas as instâncias, pois ele ajusta dinamicamente os parâmetros e pode realizar mais iterações para encontrar uma solução melhor.
- Variação entre Instâncias:
  - As instâncias 6, 8, 9 e 10 apresentam tempos de execução significativamente maiores, especialmente para o algoritmo Reativo com Lista. Isso pode indicar que essas instâncias são mais complexas ou têm um número maior de nós e arestas.
  - Por exemplo, na instância 10, o tempo de execução para o algoritmo Reativo com Lista é 1221,28 segundos, o que é muito maior do que os tempos das outras instâncias.

## 6. CONCLUSÃO

- Eficiência: A estrutura de Matriz é mais eficiente para o grafo em questão, especialmente para algoritmos mais complexos como o Reativo. Escolha do Algoritmo: O algoritmo Guloso é o mais rápido, mas pode não fornecer a melhor solução. O algoritmo Randomizado oferece um equilíbrio entre tempo de execução e qualidade da solução. O algoritmo Reativo, embora mais lento, pode ser útil para encontrar soluções de melhor qualidade em grafos complexos.
- Complexidade das Instâncias: Instâncias maiores ou mais complexas (como 6, 8, 9 e 10) requerem mais tempo de execução, especialmente com algoritmos mais sofisticados e estruturas de dados menos eficientes para o grafo em questão. Essa análise sugere que a escolha da estrutura de dados e do algoritmo deve ser feita com base no tamanho e na densidade do grafo, bem como na necessidade de balanceamento entre tempo de execução e qualidade da solução.