PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Trabalho Prático N. 02*

Teoria dos Grafos e Computabilidade

Rodrigo Drummond Macedo[1]
Henrique Resende Lara[2]
Tiago Lascasas Antunes[3]

Resumo

Este trabalho aborda a implementação e análise de duas abordagens para a resolução do problema dos k-centros: uma abordagem exata, que utiliza combinações para determinar a solução ótima, e uma abordagem aproximada, baseada em um algoritmo guloso. Os experimentos foram realizados com 40 grafos da OR-Library, variando de 100 a 900 vértices, com o objetivo de comparar a eficiência e eficácia das duas abordagens. A análise revelou que o método guloso é significativamente mais eficiente em termos de tempo de execução, enquanto o método exato fornece resultados ótimos, mas com custo computacional elevado.

Palavras-chave: Grafos. K-centros. Algoritmo guloso. Algoritmo exato.

*Artigo apresentado ao professor da disciplina Teoria de Grafos e Computabilidade, como pré-requisito para a entrega do trabalho prático número 2.

[1]Aluno do Programa de Graduação em Ciência da Computação, Brasil – rdmacedo@sga.pucminas.br.
[2]Aluno do Programa de Graduação em Ciência da Computação, Brasil – henrique.lara@sga.pucminas.br.
[3]Aluno do Programa de Graduação em Ciência da Computação, Brasil – tiago.antunes.1435526@sga.pucminas.br.

Abstract

This paper discusses the implementation and analysis of two approaches for solving the k-center problem: an exact approach, which uses combinations to determine the optimal solution, and an approximate approach, based on a greedy algorithm. Experiments were conducted with 40 graphs from the OR-Library, ranging from 100 to 900 vertices, to compare the efficiency and effectiveness of the two approaches. The analysis revealed that the greedy method is significantly more efficient in terms of execution time, while the exact method provides optimal results but with a high computational cost.

Keywords: Graphs. K-centers. Greedy algorithm. Exact algorithm.

1 INTRODUÇÃO

O problema dos k-centros é uma tarefa clássica da análise de dados, frequentemente associada a técnicas de clustering e à localização de facilidades. Dado um grafo completo ponderado, no qual as arestas indicam distâncias entre vértices, e um inteiro positivo k, o objetivo é selecionar k vértices, denominados centros, de modo a minimizar a maior distância de qualquer vértice do grafo ao centro mais próximo. Essa maior distância é chamada de raio da solução.

Embora a solução exata do problema seja inviável para instâncias de grande porte devido à sua complexidade computacional, abordagens aproximadas permitem obter soluções aceitáveis em tempo reduzido. Neste trabalho, foram implementados e comparados dois métodos distintos para resolução do problema:

- Uma abordagem exata, que utiliza a geração de combinações de k centros para encontrar a solução ótima.
- Uma abordagem aproximada, baseada em um algoritmo guloso para determinar os centros de forma iterativa.

As implementações foram testadas com as 40 instâncias disponíveis na OR-Library(BEASLEY, 1991), que variam em tamanho de 100 a 900 vértices, além de diferentes configurações de k. Este relatório apresenta os detalhes das implementações, os experimentos realizados e uma análise comparativa entre as abordagens em termos de eficiência e eficácia.

Como fontes de pesquisa foram usados (GeeksforGeeks, 2024) e ferramentas de IA, como (OPENAI, 2024), também usado para correção de erros ortográficos neste artigo.

2 DESENVOLVIMENTO

Nesta seção, são apresentados os métodos implementados para a resolução do problema dos k-centros, incluindo a construção do grafo e as especificidades de cada algoritmo.

2.1 Implementação do Grafo

Para a construção do grafo, foi utilizada uma matriz de adjacência, na qual cada posição (i, j) armazena o custo da aresta entre os vértices i e j. Inicialmente, o grafo é lido a partir dos arquivos da OR-Library, que especificam o número de vértices, arestas e seus respectivos pesos. Caso alguma aresta não esteja definida, o custo é assumido como infinito (∞).

2.1.1 Função Floyd-Warshall (`floydWarshall`)

A função `floydWarshall` é utilizada para calcular todas as distâncias mínimas entre pares de vértices, assegurando a validade das distâncias e o cumprimento da desigualdade triangular. Isso é essencial para transformar o grafo original em um grafo completo ponderado. Essa etapa tem complexidade $O(V^3)$, sendo adequada para o tamanho das instâncias testadas.

2.2 Algoritmo Exato

O método exato realiza uma busca completa por todas as combinações possíveis de k vértices para identificar a combinação que minimiza o raio. Embora garantida a solução ótima, sua complexidade combinatorial torna-o impraticável para instâncias maiores.

2.2.1 Função de Geração de Combinações

A geração de combinações dos k vértices é implementada implicitamente dentro da função `forçaBrutaKcentros`, utilizando uma abordagem iterativa para explorar todas as possibilidades de forma eficiente. A cada nova combinação, o raio associado é calculado.

2.2.2 Função de Cálculo do Raio

A função `calcularRaio` avalia a qualidade de cada solução candidata calculando o raio como a maior distância entre qualquer vértice e o centro mais próximo, considerando a combinação atual de centros.

2.2.3 Complexidade

A complexidade do algoritmo exato é $O(Vk \cdot V)$, devido à necessidade de avaliar todas as combinações possíveis e calcular o raio para cada uma. Isso limita sua aplicação a instâncias pequenas, como demonstrado nos experimentos.

2.3 Algoritmo Aproximado

O algoritmo guloso aproxima a solução selecionando os k centros de forma iterativa, priorizando vértices que maximizam a distância ao centro mais próximo já selecionado. Embora

não garanta a solução ótima, apresenta tempos de execução significativamente menores.

2.3.1 Função de Inicialização

A função `initialize` define o número de centros (k) a serem selecionados e prepara a estrutura para iniciar o processo.

2.3.2 Função de Seleção Iterativa de Centros

A função `computeCenters` é o núcleo do algoritmo. A cada iteração:

1. Identifica o vértice mais distante do conjunto atual de centros (`maxindex`).
2. Atualiza as distâncias mínimas entre os vértices e os centros.
3. Seleciona o próximo centro até que k centros sejam definidos.

2.3.3 Cálculo do Raio Final

A função `getRadius` retorna o raio calculado após a seleção dos k centros, definido como a maior distância de um vértice ao centro mais próximo.

2.3.4 Complexidade

A complexidade do algoritmo aproximado é $O(k \cdot V^2)$, sendo consideravelmente mais eficiente que o método exato, especialmente para instâncias grandes.

3 EXPERIMENTOS

Os experimentos foram realizados utilizando as 40 instâncias da OR-Library. Os resultados incluem os raios das soluções obtidas e os tempos de execução para ambos os métodos. As tabelas a seguir ilustram os dados para o algoritmo aproximado.

3.1 Tabela e Gráficos

Tabela 1 – Resultados do algoritmo aproximado

Instância	Raio	Tempo de Execução (ms)
pmed1	186	7
pmed2	131	3
pmed3	154	3
pmed4	114	4
pmed5	71	1
...	...	...
pmed40	21	355

Para o método exato, apenas a instância $pmed1$ foi executada, devido ao alto tempo de execução:

- Instância $pmed1$: Raio = 127, Tempo de Execução = 20503 ms.

Figura 1 – Comparação de raios obtidos pelos algoritmos exato e aproximado.

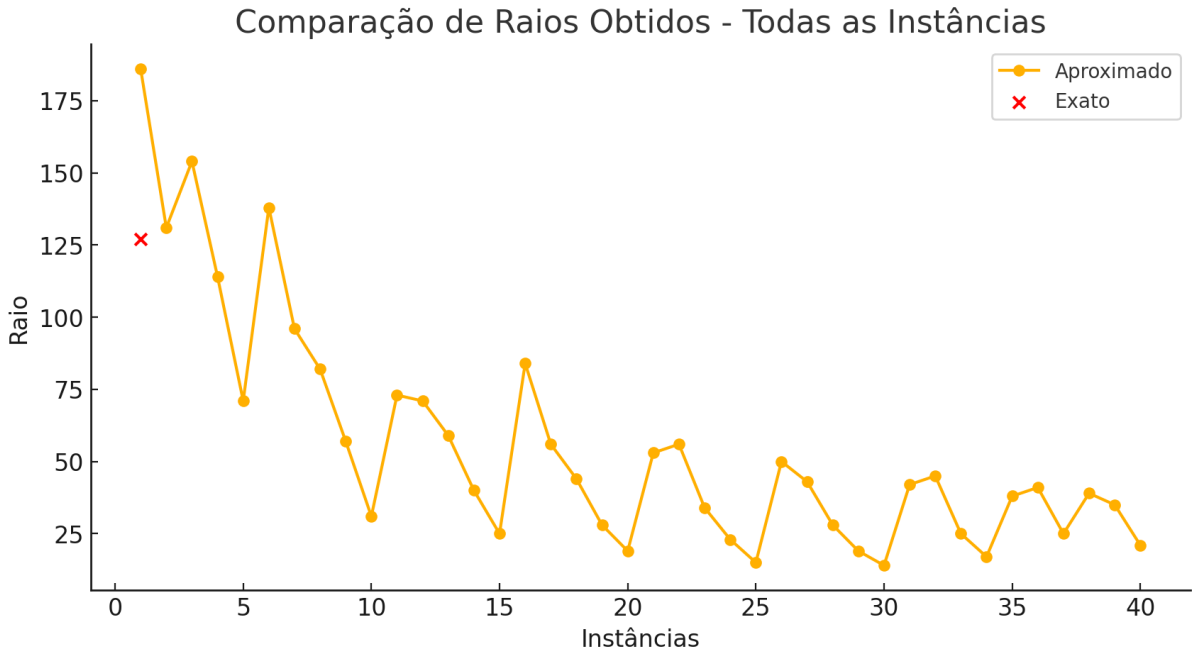


Figura 2 – Comparação de tempos de execução entre os algoritmos exato e aproximado (escala logarítmica).

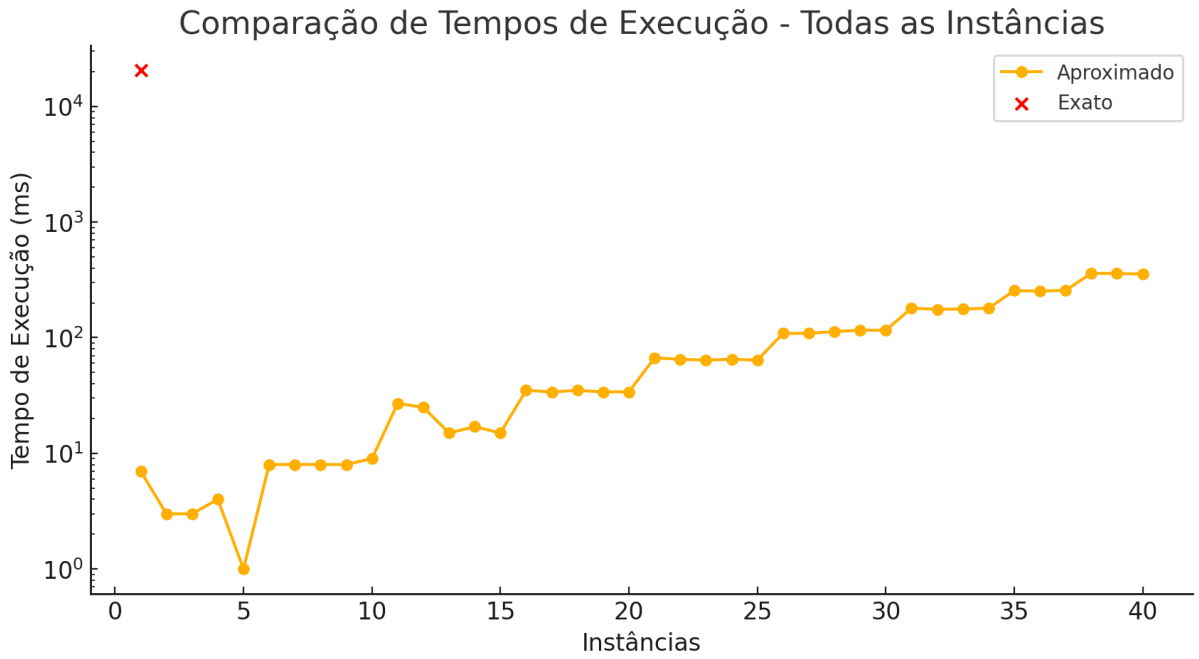
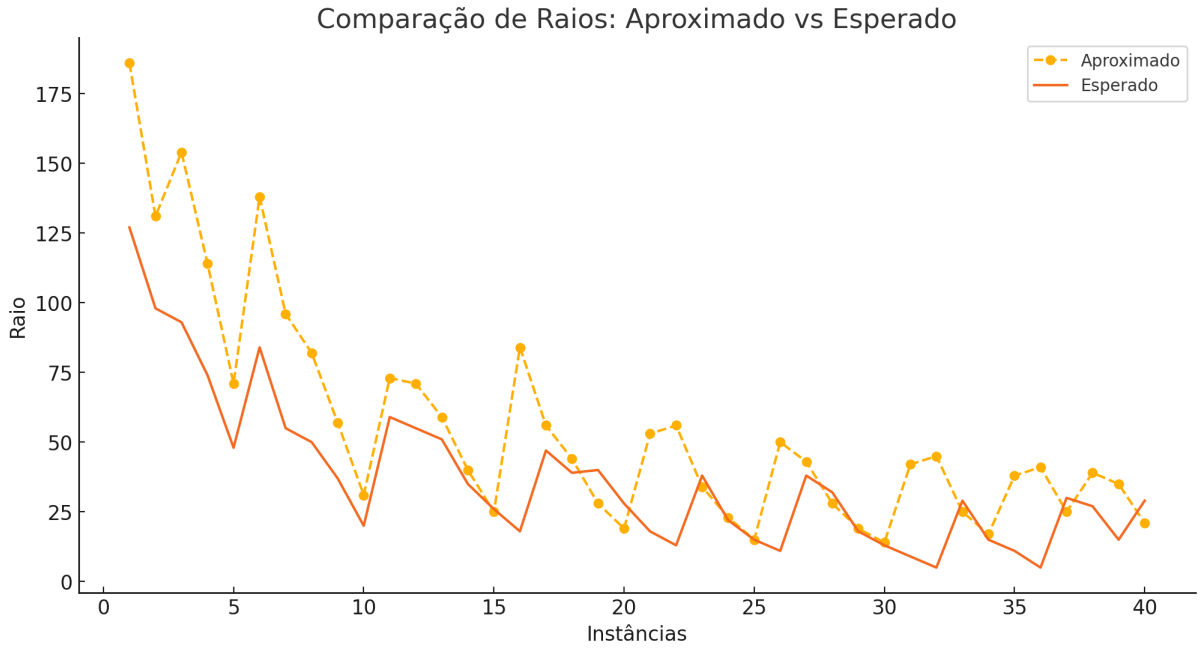


Figura 3 – Comparação De Raios: Aproximado Vs Esperado



4 RESULTADOS E ANÁLISE

Raios Obtidos: O algoritmo exato apresentou o menor raio (127) para a instância analisada, confirmando sua precisão. O algoritmo aproximado, embora tenha fornecido raios maiores para a mesma instância, demonstrou resultados consistentes e aceitáveis em todas as 40

instâncias testadas. A redução gradual dos raios nas instâncias maiores reflete a eficácia do aproximado em encontrar soluções práticas próximas do ótimo.

Tempos de Execução: O algoritmo exato teve um tempo de execução muito elevado (20.503 ms) para a instância pmed1, tornando-o inviável para instâncias maiores. Em contrapartida, o algoritmo aproximado demonstrou tempos de execução muito mais baixos, variando de 1 ms a 355 ms em todas as instâncias, com um crescimento gradual e previsível conforme o tamanho dos grafos aumenta.

Resultado: A análise reforça que o algoritmo aproximado é altamente eficiente e adequado para aplicações práticas envolvendo grandes instâncias. Apesar de não fornecer a solução ótima, sua precisão é aceitável e seu desempenho é amplamente superior ao exato. O algoritmo exato, embora preciso, é mais indicado para instâncias pequenas, onde a obtenção da solução ótima é imprescindível.

5 CONCLUSÃO

Conclui-se que o algoritmo aproximado é mais adequado para resolver o problema dos k-centros em instâncias grandes, devido à sua eficiência em termos de tempo de execução. Já o algoritmo exato é limitado a instâncias pequenas, onde o custo computacional ainda é administrável. Dessa forma, existe um trade-off em relação a tempo de execução e precisão, o qual fica a cargo do usuário escolher a opção que melhor lhe sirva.

REFERÊNCIAS

BEASLEY, J.E. Or-library: Distributing test problems by electronic mail. **Journal of the Operational Research Society**, Springer, v. 41, n. 11, p. 1069–1072, 1991. Accessed: 2024-10-06.

GeeksforGeeks. **Geeks for Geeks Website**. 2024. <https://www.geeksforgeeks.org/>. Accessed: 2024-10-06.

OPENAI. **ChatGPT: Language Model**. 2024. <https://openai.com/chatgpt>. Accessed: 2024-10-06.