PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Graduação em Ciência da Computação

Bernardo D Ávila Rodrigues Bartholomeu
Gabriel Azevedo Fernandes
Pedro Henrique Moreira

TRABALHO PRÁTICO N.02

Belo Horizonte
2024

RESUMO

O trabalho prático 2 da disciplina de Teoria dos Grafos e Computabilidade no semestre 2024/2 aborda a implementação e análise de duas abordagens para o problema dos k-centros: uma solução exata, voltada para instâncias pequenas, e uma solução aproximada, otimizada para instâncias maiores. Utilizando dados da OR-Library, as implementações foram avaliadas em termos de eficiência e precisão, considerando o impacto do aumento do tamanho das instâncias. Os resultados destacam os trade-offs entre desempenho e precisão, oferecendo insights sobre a aplicabilidade prática de cada abordagem.

Palavras-chave: Problema dos k-centros, Clustering, Otimização Combinatória, Análise de Grafos, Métodos Exatos e Aproximados, Desempenho Computacional, OR-Library, Análise Comparativa

Sumário

Introdução

O problema dos k-centros é essencial na análise de dados e nas técnicas de clustering, com aplicações significativas que variam de otimização de redes a segmentação de mercado. Neste contexto, o desafio é identificar um conjunto de k vértices, denominados centros, em um grafo completo com arestas ponderadas, de modo que a distância máxima entre qualquer vértice e o centro mais próximo seja minimizada. Este trabalho visa comparar duas metodologias distintas: uma abordagem exata, que assegura a solução ótima mas pode ser inviável computacionalmente para grandes conjuntos de dados, e uma abordagem aproximada, que oferece rapidez e praticidade com uma precisão aceitável para cenários práticos.

Metodologia

Nesta seção, apresentamos as duas abordagens utilizadas para resolver o problema dos k-centros: o algoritmo aproximado KMeans e a abordagem exata Brute Force. Além disso, explicamos como o algoritmo Floyd-Warshall foi utilizado como suporte para o pré-processamento das distâncias entre os vértices do grafo.

KMeans: Abordagem Aproximada

O KMeans é uma solução heurística para o problema dos k-centros, buscando uma boa aproximação para a solução ótima de forma eficiente, especialmente em grafos grandes. Essa abordagem consiste em uma sequência iterativa de três etapas principais:

1. Inicialização dos centróides: Os centróides, que representam os k centros iniciais, são escolhidos aleatoriamente entre os vértices do grafo. Para garantir que não haja repetições, cada novo centróide é verificado antes de ser adicionado à lista.

```
Input: n (number of vertices), k (number of clusters)
Output: List of initial centroids
Initialize an empty list of centroids
while size of centroid list < k do:
    Select a random vertex between 0 and n-1
    if vertex is not in the centroid list:
        Add vertex to the centroid list
return centroid list
```

2. Atribuição de clusters e atualização dos centróides: Cada vértice do grafo é atribuído ao centróide mais próximo, formando clusters. Em seguida, os centróides são recalculados com base nos vértices mais centrais de cada cluster. Esse processo é repetido até que os centróides parem de mudar, indicando convergência.

```
Input: Distance matrix G, list of centroids
Output: Updated clusters and new centroids
Initialize an empty list of clusters
for vertex v in G:
    Find the closest centroid to v
    Assign v to the corresponding cluster
for cluster in clusters:
    Recalculate centroid by minimizing the average distance
    to other vertices in the cluster
return updated clusters, list of new centroids
```

3. Cálculo do raio: O raio da solução é calculado como a maior distância entre qualquer vértice e o centróide do cluster ao qual ele pertence. Durante as iterações, o algoritmo tenta minimizar esse valor, resultando em uma solução eficiente.

```
Input: Distance matrix G, clusters
Output: Radius of the solution
Initialize maxRadius = 0
for cluster C in clusters:
    for vertex v in C:
        Calculate the distance from v to the centroid of C
        Update maxRadius if the distance is greater
return maxRadius
```

Vantagens e Limitações: O KMeans é rápido e funciona bem em grafos maiores devido à sua capacidade de realizar múltiplas iterações em curto tempo. No entanto, sua precisão depende da inicialização dos centróides, o que pode levar a resultados menos consistentes. Além disso, como se trata de uma heurística, não há garantia de alcançar a solução ótima.

Brute Force: Abordagem Exata

O algoritmo Brute Force é uma abordagem que garante encontrar a solução ótima para o problema dos k-centros. Ele realiza uma busca exaustiva em todas as possíveis combinações de k vértices do grafo, avaliando cada configuração para determinar a melhor solução. No entanto, seu custo computacional cresce exponencialmente com o tamanho do grafo, o que limita sua aplicação a instâncias menores. Ele segue três etapas principais:

1. Geração de combinações: O algoritmo gera todas as possíveis combinações de k vértices entre os n vértices do grafo. Isso é feito utilizando uma abordagem recursiva, que explora todas as configurações possíveis. A quantidade de combinações cresce de acordo com o coeficiente binomial $\binom{n}{k}$, tornando o processo inviável para grafos grandes.

```
Input: Distance matrix G, number of vertices n, number of centers k
Output: Best combination of centers with the smallest radius
Define currentCombination as an empty list
Define bestRadius as INF
Define bestCenters as NULL

Function generateCombinations(start):
    if size of currentCombination == k:
        radius = calculateRadius(G, currentCombination)
        if radius < bestRadius:
            bestRadius = radius
            bestCenters = copy of currentCombination
        return

    for i from start to n:
        Add i to currentCombination
        generateCombinations(i + 1)
        Remove last element from currentCombination
```

2. Avaliação do raio: Para cada combinação de k vértices gerada, calcula-se o raio correspondente. O raio é definido como a maior distância entre qualquer vértice do grafo e o centro mais próximo. Essa etapa é essencial para determinar qual combinação oferece a melhor solução.

```
Input: Distance matrix G, list of centers
Output: Radius of the current combination
Function calculateRadius(centers):
    maxRadius = 0
    for each vertex i in G:
        minDist = INF
        for each center in centers:
            minDist = min(minDist, G[i][center])
        maxRadius = max(maxRadius, minDist)
    return maxRadius
```

3. Seleção da melhor combinação: Após avaliar todas as combinações, o algoritmo seleciona a configuração de k centros que apresenta o menor raio. Essa solução é considerada ótima, garantindo a minimização da distância máxima entre os vértices e seus centros.

Vantagens e Limitações: A principal vantagem do Brute Force é a garantia de encontrar a solução ótima para o problema dos k-centros. Isso o torna ideal para aplicações em instâncias pequenas ou como método de validação para outros algoritmos aproximados. No entanto, devido à sua complexidade $O(\binom{n}{k})$, o algoritmo se torna impraticável para grafos grandes, onde o tempo de execução cresce rapidamente com o número de vértices.

Resultados Práticos: Nos experimentos realizados, o método Brute Force levou 77.489 ms para resolver a instância PMED1, fornecendo um raio de 127. Esses resultados confirmam sua eficácia para instâncias pequenas, mas reforçam sua limitação para aplicações em escalas maiores.

Floyd-Warshall: Algoritmo Auxiliar

O algoritmo Floyd-Warshall foi utilizado como um passo preliminar antes da execução dos métodos principais. Ele tem como objetivo calcular as menores distâncias entre todos os pares de vértices no grafo.

Durante esse pré-processamento, o algoritmo verifica se um caminho indireto entre dois vértices pode ser mais curto do que o caminho direto. Caso positivo, ele atualiza a distância correspondente. O resultado final é uma matriz completa de distâncias mínimas, essencial para a execução precisa do KMeans e do Brute Force.

Papel no Contexto Geral: Embora o Floyd-Warshall não resolva diretamente o problema dos k-centros, ele prepara o grafo para que as distâncias sejam consistentes e otimizadas, facilitando a execução dos outros métodos. Com uma complexidade de $O(n^3)$, ele é eficiente o suficiente para ser aplicado em grafos moderados, mesmo como etapa auxiliar.

Essa estrutura garante que tanto a abordagem heurística quanto a abordagem exata sejam compreendidas, enquanto o papel do Floyd-Warshall é destacado como fundamental para o sucesso dos métodos principais.

Testes e Resultados

Os experimentos foram realizados utilizando as instâncias da OR-Library, amplamente reconhecida por oferecer benchmarks confiáveis em problemas de otimização. O objetivo principal foi avaliar o desempenho de cada método em termos de eficiência e precisão, considerando diferentes tamanhos de instâncias.

O método **Brute Force** foi limitado a uma única instância, PMED1, devido à sua complexidade computacional exponencial, que torna impraticável sua aplicação em grafos maiores. Já o método **KMeans** foi testado em todas as instâncias disponíveis, demonstrando sua viabilidade em escalas maiores graças à sua natureza heurística.

As métricas de avaliação incluem:

- **Tempo de execução**: Representa o custo computacional do método, medido em milissegundos.
- **Raio da solução**: Indica a qualidade da solução, definido como a maior distância entre um vértice e o centro mais próximo.

Para assegurar resultados precisos, foi aplicado o algoritmo Floyd-Warshall no pré-processamento, gerando a matriz de distâncias mínimas entre todos os pares de vértices. Os resultados destacam as diferenças fundamentais entre a abordagem exata e a heurística, tanto em desempenho quanto em qualidade da solução.

Resultados do Método Brute Force

O método Brute Force foi executado apenas para a instância PMED1 devido ao alto custo computacional. Os resultados estão apresentados na Tabela 1.

Tabela 1 – Resultados do Método Brute Force

Instância(pmed)	Tempo de Execução (ms)	Valor do Raio
1	77489,80	127.0

Resultados do Método KMeans

O método KMeans foi executado para todas as instâncias disponíveis. Os resultados estão apresentados na Tabela.

Tabela 2 – Resultados Combinados do Método KMeans

Instância (pmed)	Tempo de Execução (ms)	Valor do Raio
1	4,10	156.0
2	1,45	129.0
3	2,45	186.0
4	3,15	96.0
5	1,51	90.0
6	2,02	113.0
7	2,69	96.0
8	1,98	109.0
9	1,04	91.0
10	2,27	68.0
11	3,75	76.0
12	1,12	105.0
13	0,74	77.0
14	0,63	75.0
15	1,01	57.0
16	1,34	55.0
17	10,22	61.5
18	10,69	61.5
19	15,85	45.4
20	23,08	44.8
21	21,66	60.3
22	15,29	68.9
23	15,80	45.7
24	23,70	40.8
25	36,48	45.8
26	37,63	53.3
27	20,72	55.2
28	20,46	60.0
29	32,46	38.6
30	48,93	38.6
31	35,35	46.8
32	27,58	87.6
33	26,85	35.4
34	43,55	47.0
35	52,21	45.7
36	30,21	55.6
37	33,47	40.0
38	58,57	52.0
39	39,26	88.6
40	43,19	30.9

Conclusão

Com base nos resultados obtidos, podemos concluir que, apesar do método de força bruta garantir a solução ótima para o problema dos k-centros, sua aplicação é inviável para instâncias maiores devido ao custo computacional elevado. No teste realizado com a instância PMED1, o Brute Force levou 77.489 ms para produzir um raio de 127.0, demonstrando sua limitação para problemas de maior escala.

Por outro lado, o método KMeans apresentou tempos de execução variando entre 0,63 ms e 58,57 ms, com raios entre 30.9 e 186.0. Comparando os dois métodos, o KMeans mais lento foi 1.323 vezes mais rápido que o Brute Force, enquanto o KMeans mais rápido alcançou um desempenho aproximadamente 123.015 vezes superior. Apesar de não garantir a solução ótima, o KMeans obteve resultados satisfatórios em termos de precisão, com o menor raio encontrado sendo apenas 24,3% do valor do Brute Force (30.9 contra 127.0).

Os resultados indicam que, embora o Brute Force seja adequado para instâncias pequenas onde a precisão é crucial, o KMeans se destaca como uma solução prática e eficiente para aplicações em grafos maiores, equilibrando tempo de execução e qualidade da solução. Isso reforça a importância de adaptar a escolha do método às limitações computacionais e às necessidades específicas do problema.

Referências Bibliográficas

GitHub: https://github.com/pedrohenrimoreira/Graphs-TP02.git

Instâncias Utilizadas: https://people.brunel.ac.uk/ mastjjb/jeb/orlib/files/

OR-Library: https://people.brunel.ac.uk/ mastjjb/jeb/info.html

Floyd Wharsall (Geeks for Geeks): https://people.brunel.ac.uk/ mastjjb/-jeb/info.html