Trabalho Prático 02: Problema dos k-Centros

Arthur Costa Serra Negra
Gabriel Costa Vianna

Junho de 2025

Resumo

Este relatório descreve as implementações e os experimentos realizados para a resolução do problema dos k-centros, conforme o enunciado do TP02 da disciplina de Teoria dos Grafos e Computabilidade. Foram implementados dois métodos: (i) uma heurística aproximada (algoritmo de Gonzalez) e (ii) uma solução exata por força bruta. Testamos as implementações nas 40 instâncias recomendadas (pmed1..pmed40). O método aproximado foi executado em todas as 40 instâncias; o método exato foi possível apenas para a instância pmed1 (demais instâncias ultrapassaram o limite combinatório).

1 Introdução

O problema dos k-centros busca selecionar até k vértices (centros) de um grafo métrico de modo a minimizar o maior custo (distância) de um vértice até o centro mais próximo — esse valor é chamado raio da solução. O enunciado do trabalho e as instâncias foram obtidos da OR-Library, conforme especificado no enunciado do TP.[1]

2 Implementação

2.1 Leitura e representação

As instâncias foram carregadas e representadas por uma matriz de distâncias. Aplicou-se o algoritmo de Floyd–Warshall para garantir que distâncias mínimas entre todos pares estivessem disponíveis antes da execução dos solvers.

2.2 Solver aproximado (Gonzalez)

Implementamos a heurística gulosa de Gonzalez: inicia-se com um vértice arbitrário (0) como primeiro centro e, iterativamente, escolhe-se o vértice cuja distância mínima ao conjunto de centros já selecionados é máxima. A complexidade é $O(nk)$ por execução.

Listing 1: Trecho do ApproximateSolver.java (resumido)

```
// inicializa com vertice 0
centers.add(0);
```

[1] TGC TP02 - enunciado.

```
for (int iter = 1; iter < k; ++iter) {
  // para cada v rtice , calcula a dist ncia m nima aos centros
  // escolhe o v rtice com maior dist ncia m nima
}
// calcula raio final
```

2.3 Solver exato (força bruta)

A solução exata testa todas as combinações $C(n, k)$ e calcula o raio. Para evitar estouro de tempo/memória, desabilitamos a execução quando o número de combinações excede um limite (implementado como verificação de overflow e limite prático). No experimento, apenas pmed1 foi resolvida exatamente — para outras instâncias $C(n, k)$ era impraticável.

Listing 2: Trecho do ExactSolver.java (resumido)

```
// calcula combinacoes C(n,k) com checagem de overflow
if (totalComb > LIMITE_COMBINACOES) {
  // desabilita exato
}
// gera combinacoes recursivamente e atualiza melhor raio
```

3 Experimentos

As 40 instâncias pmed1..pmed40 foram executadas com o solver aproximado. A tabela 1 apresenta os valores de $|V|$, k, o raio ótimo (conforme Tabela do enunciado), o raio obtido pelo método aproximado, a diferença percentual e o tempo de execução do método aproximado em milissegundos. Para o método exato, apenas pmed1 foi executada (resultado incluso e discutido separadamente).

Tabela 1: Comparacção entre raio ótimo (enunciado) e raio obtido pelo método aproximado.

Instância	$\|V\|$	k	Raio ótimo	Raio aprox.	Dif. (%)	Tempo (ms)
n						
pmed1.txt	100	5	127	186	46.5	0.4552
n pmed2.txt	100	10	98	131	33.7	0.8771
n pmed3.txt	100	10	93	154	65.6	0.0
n pmed4.txt	100	20	74	114	54.1	0.0
n pmed5.txt	100	33	48	71	47.9	1.0
n pmed6.txt	200	5	84	138	64.3	0.0
n pmed7.txt	200	10	64	96	50.0	0.0
n pmed8.txt	200	20	55	82	49.1	1.0
n pmed9.txt	200	40	37	57	54.1	0.0
n pmed10.txt	200	67	20	31	55.0	20.0
n pmed11.txt	300	5	59	73	23.7	0.0
n pmed12.txt	300	10	51	71	39.2	2.0
n pmed13.txt	300	30	35	59	68.6	0.0
n pmed14.txt	300	60	26	40	53.8	2.0

Instância	$\|V\|$	k	Raio ótimo	Raio aprox.	Dif. (%)	Tempo (ms)
n pmed15.txt	300	100	18	25	38.9	2.0
n pmed16.txt	400	5	47	84	78.7	0.0
n pmed17.txt	400	10	39	56	43.6	0.0
n pmed18.txt	400	40	28	44	57.1	1.0
n pmed19.txt	400	80	18	28	55.6	3.0
n pmed20.txt	400	133	13	19	46.2	7.0
n pmed21.txt	500	5	40	53	32.5	0.0
n pmed22.txt	500	10	38	56	47.4	0.0
n pmed23.txt	500	50	22	34	54.5	3.0
n pmed24.txt	500	100	15	23	53.3	6.0
n pmed25.txt	500	167	11	15	36.4	18.0
n pmed26.txt	600	5	38	50	31.6	0.0
n pmed27.txt	600	10	32	43	34.4	0.0
n pmed28.txt	600	60	18	28	55.6	4.0
n pmed29.txt	600	120	13	19	46.2	7.0
n pmed30.txt	600	200	9	14	55.6	15.0
n pmed31.txt	700	5	30	42	40.0	0.0
n pmed32.txt	700	10	29	45	55.2	0.0
n pmed33.txt	700	70	15	25	66.7	5.0
n pmed34.txt	700	140	11	17	54.5	16.0
n pmed35.txt	800	5	30	38	26.7	0.0
n pmed36.txt	800	10	27	41	51.9	59.0360
n pmed37.txt	800	80	15	25	66.7	8.0
n pmed38.txt	900	5	29	39	34.5	0.0
n pmed39.txt	900	10	23	35	52.2	0.0
n pmed40.txt	900	90	13	21	61.5	20.0

4 Resultados do método exato

Somente a instância pmed1 (100 vértices, $k = 5$) foi possível de ser resolvida por força bruta dentro do limite definido. O resultado obtido foi:

- Raio exato (ótimo): 127
- Centros (0-indexado): [4, 8, 56, 62, 77]
- Tempo de execução (força bruta): 65847.3358 ms (65.85 s)

A diferença entre o método aproximado e o exato para pmed1 é de 46.5%, refletindo a conhecida perda de qualidade da heurística em algumas instâncias; entretanto, a heurística é extremamente mais rápida.

5 Análise

5.1 Qualidade das soluções

A heurística de Gonzalez tem garantia teórica de fator 2, ou seja, $Raio_{aprox} \leq 2 \cdot OPT$. Na prática, observamos diferenças percentuais variando entre aproximadamente 23% até mais de 78% (pmed16). A diferença média observada no conjunto completo foi de aproximadamente 48.7%, o que coincide com o resultado resumido no relatório de referência.

5.2 Desempenho

O tempo de execução do método aproximado foi sempre muito baixo (milissegundos), mesmo para instâncias com 900 vértices. O método exato tem complexidade combinatória e explodiu rapidamente: para várias instâncias o número de combinações foi estimado acima do limite prático e o solver exato foi desabilitado.

5.3 Discussão

A heurística é apropriada quando se exige solução rápida em instâncias grandes, ao custo de precisão. Para aplicações em que o raio ótimo é crítico, estratégias híbridas (heurística seguida de refinamentos locais ou métodos de busca meta-heurística) podem reduzir a diferença observada.

6 Conclusão

Implementamos e avaliamos duas abordagens para o problema dos k-centros. A heurística de Gonzalez mostrou-se eficiente e escalável, embora com perda de qualidade em relação ao ótimo. O método exato confirma a validade da heurística em pequenas instâncias, mas é impraticável na maioria dos casos devido ao crescimento combinatório.

7 Códigos-fonte

Os códigos principais (arquivos Java) entregues junto com este relatório são:

- Main.java
- InstanceReader.java
- ApproximateSolver.java
- ExactSolver.java
- Demais utilitários (leitura e parsing das instâncias)

Referências

1. Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. Theoretical Computer Science, 38, 293–263.

2. Enunciado do TP02 - Teoria dos Grafos e Computabilidade (PUC-MG).[2]

[2]Arquivo fornecido pelo professor e incluído no pacote de entrega.