Teoria dos Grafos

IMPLEMENTAÇÃO DE GRAFOS:
ANÁLISE DA DISTRIBUIÇÃO DE COMPONENTES CONEXOS
EM FUNÇÃO DE LIMIARES NA CONSTRUÇÃO DAS ARESTAS

Alunos: Luciano Kohler e Matheus E. G. Santana
Professor: Gilmário Barbosa dos Santos

Joinville - 2025

SUMÁRIO

1 INTRODUÇÃO

Um grafo é uma estrutura de dados primordial na área da programação, a mesma é representada por um conjunto de vértices que podem ou não estarem conectados via uma aresta, indicando uma conexão lógica entre os dois elementos. Usualmente, grafos são representados a partir de uma tripla ordenada (N, A, g), onde 'N' é um conjunto não vazio de nós (vértices), 'A' é um conjunto de arcos (arestas), que correspondem a pares não ordenados, e 'g' é uma função que associa a cada arco um par não ordenado x-y de nós, chamados de extremidades de 'a'.

Neste relatório, também será usado o conceito de "matriz de adjacência", que se refere a um método de representar um grafo computacionalmente. Essa representação ocorre por meio de uma matriz nxn, com n correspondendo ao número de vértices pertencentes ao grafo, e onde cada elemento a[i][j] corresponde à quantidade de arestas que um vértice i possui para o vértice j. Segue uma imagem para auxílio da explicação:

	v1	v2	v3	v4
v1	1	0	1	1
v2	0	0	0	1
v3	1	0	0	1
v4	1	1	1	0

Matriz de adjacência do grafo G

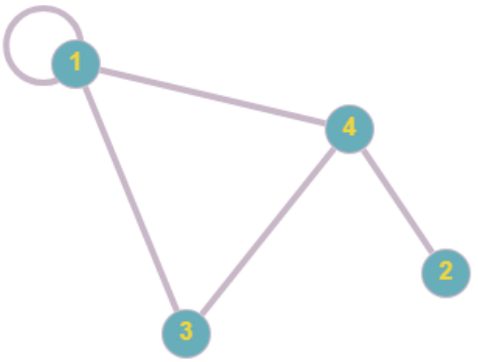

Figura 1[1]: Imagem ilustrando o grafo G

Em grafos, existe o conceito de "grupos conexos", ou "clusters", que são separações dos vértices em grupos individuais compostos por outros vértices que ocorrem geralmente quando se é trabalhado com um grande volume de nodos (nossos dados), que possuem algum tipo de correlação entre si. Uma análise desses grupos conexos, principalmente em bases de dados mais robustas, pode vir a se provar muito benéfica para encontrar otimizações e padrões de comportamento entre os vértices analisados.

2 OBJETIVOS

O intuito deste relatório é, portanto, documentar uma análise realizada sobre uma base de dados com 150 vértices disponibilizada pelo professor, onde cada nodo é representado por uma coordenada quadridimensional (X, Y, Z, W) e, por meio de um limiar/tolerância aplicada sobre uma normalização da distância euclidiana de cada vértice com outro vértice, cada um deles pode ou não estar conectado por meio deste limiar que pode variar entre 0 (nodos só se conectam se forem iguais) a 1 (qualquer nodo conecta com qualquer nodo).

3 METODOLOGIA

3.1 Recursos para o experimento

- Utilização de um notebook Dell Inspiron 15:
  - Memória RAM: 16GB
  - Processador: Intel Core i5-1235U 10 núcleos
  - Tipo de unidade de armazenamento: SSD M.1 512GB

- Compilação do código em C feita à partir do GCC versão 14.2.0 e executado via linha de comando no terminal (.\main.exe)

- Bibliotecas utilizadas:
  - stdio.h e stdlib.h (padrões de qualquer código C
  - string.h para manipulação de strings e leitura de dados
  - math.h para o cálculo das distâncias euclidianas

- Implementação da estrutura de grafo contendo:
  - Coordenadas (quádrupla)
  - Distâncias de nodo a nodo
  - Distâncias normalizadas (variando de 0 a 1)
  - Matriz de adjacência para cada limiar analisado (0.0, 0.3, 0.5, 0.9)

- Utilização de leitura e escrita de dados em arquivos CSV, contendo na pasta input o **dataset.csv** com as coordenadas dos 150 vértices, e, na pasta output, os arquivos **adjacenciasLimiar0_x.csv**, onde cada arquivo representa a matriz de adjacência com um limiar específico (representado pelo “x”) do arquivo

3.2 Procedimento experimental

O processo de análise foi inteiramente feito em um único arquivo C chamado **main.c**, onde nele, foram seguidos os seguintes passos:

1. **Declaração de variáveis**, onde todas as variáveis necessárias para o experimento foram declaradas em relação ao tamanho da amostra (150).
2. **Leitura do arquivo de entrada**, onde por meio do dataset.csv, foram recebidas e salvas as coordenadas de cada nodo.
3. **Cálculo de distâncias e procura da distância máxima e mínima**, onde para cada par de nodos, foi calculada a distância euclidiana deles e, se a distância entre ambos for menor que a mínima ou maior que a máxima, este valor também é salvo para uso posterior no passo de cálculo de distâncias normalizadas. A fórmula[2] utilizada para o cálculo de distâncias euclidiana é a seguinte:

$$d = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$

Figura 2: Fórmula para o cálculo da distância euclidiana de dois pontos em uma dimensão n qualquer.

4. **Cálculo de distâncias normalizadas**, onde por meio da fórmula de normalização de um conjunto de dados[3], foram normalizadas as distâncias, fazendo com que as mesmas variem de 0 a 1. A fórmula utilizada para o cálculo das distâncias normalizadas é a seguinte:

$$\hat{X} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Figura 3: Fórmula para o cálculo da normalização de um conjunto de dados qualquer.

5. **Produção das matrizes de adjacência para cada limiar alvo**, onde são criados quatro arquivos de output, cada um contendo as matrizes de adjacência do conjunto para diferentes limiares.
6. **Escolha de limiar para análise de clusters**, aqui, é requisitado ao usuário escolher um limiar calculado (dentre os quatro já citados), para a análise da quantidade de clusters encontrados no grafo.
7. **Busca da quantidade de clusters no grafo**, onde por fim, é utilizado o algoritmo de DFS[4] (Depth First Search) para encontrar quantos nodos estão interligados em cada cluster, assim, retornando quantos clusters existem e a quantidade de elementos em cada um.

4 RESULTADOS E DISCUSSÃO

Com os experimentos devidamente realizados e os dados depurados, foi possível notar um comportamento esperado para a quantidade de clusters no grafo de limiar 0.0, onde, como pode ser visto pela figura 4, a existência de clusters com vários elementos é totalmente escassa, isso provém do fato de que, pelo limiar ser nulo, apenas nodos que possuem as exatas mesmas coordenadas irão ter uma conexão entre eles, logo, pelos dados gerados, é possível ver que existem 145 nodos com coordenadas diferentes, um par de nodos com coordenadas iguais, e um trio de nodos também com mesmas coordenadas.

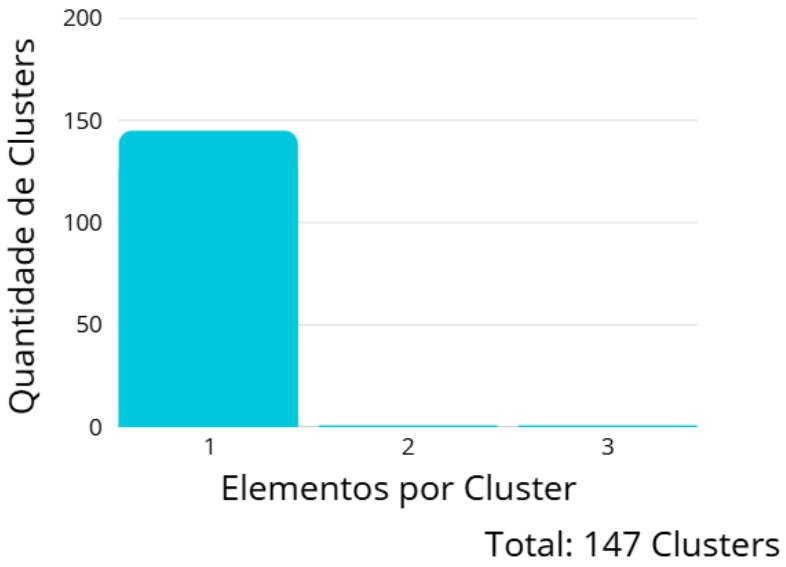

Figura 4: Distribuição de Clusters em um cenário de limiar 0.0.

Para os limiares 0.3, 0.5 e 0.9, os resultados foram os mesmos: Todos os nodos conseguiram se conectar em um único cluster que engloba todo o conjunto de vértices no grafo, como se pode ver na figura 5, há um único cluster de 150 elementos em todos os três casos testados.

Figura 5: Distribuição de Clusters nos demais cenários (limiar 0.3, 0.5 e 0.9).

5 CONCLUSÃO

É possível, após esta análise, perceber que o comportamento de um cluster em uma base de dados pequena é muito volátil, pois, com uma variação de apenas 0.3 pontos, o comportamento do grafo se alterou completamente, essa volatilidade pode se tornar ainda mais delicada se aplicada em uma base de dados mais ampla que mantenha o mesmo alcance das coordenadas, pois com mais nodos, é mais fácil haver um conjunto de nodos que sirva de “ponte” entre um cluster e outro, fundindo-os no processo..

Como é visto no comportamento das arestas em função do limiar, não temos conexão alguma com o limiar nulo, e uma conexão completa do grafo com o limiar 0.3, isso leva a crer que, pela natureza das distâncias normalizadas e dos limiares, deve haver algum limiar entre 0 e 0,3 que separe os vértices em grupos mais equilibrados. Para descobri-lo, entretanto, serão necessárias adaptações no código e na metodologia da análise para encontrar o limiar ideal para a separação homogênea dos clusters, adaptações essas que serão aplicadas em fases futuras da pesquisa.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Graph Online. *Criando um grafo a partir de uma matriz de adjacência*. **Graph Online**, [s.d.]. Disponível em: https://graphonline.top/pt/create_graph_by_matrix. Acesso em: 10 out. 2025.

[2] Vinod Chugani. *Entendendo a distância euclidiana: Da teoria à prática*. **DataCamp**, 1 de out. de 2024. Disponível em: https://www.datacamp.com/pt/tutorial/euclidean-distance. Acesso em: 10 out. 2025.

[3] Aashish Nair. *Standardization vs normalization*. **Towards Data Science**, 21 mar. 2022. Disponível em: https://.com/standardization-vs-normalization-dc81f23085e3/. Acesso em: 10 out. 2025.

[4] GEEKSFORGEEKS. *Depth First Search or DFS for a Graph*. **GeeksforGeeks**, 3 out. 2025. Disponível em: https://www.geeksforgeeks.org/dsa/depth-first-search-or-dfs-for-a-graph/. Acesso em: 10 out. 2025.