RICARDO NORIO MIYATA

EXPLORANDO A CENTRALIDADE DOS CAMINHOS MÍNIMOS EM GRAFOS.

(*versão pré-defesa, compilada em 5 de dezembro de 2021*)

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Murilo V. G. da Silva.

CURITIBA PR

2021

RESUMO

O objetivo deste trabalho é, por meio de três tipos de gráficos de entrada, enfatizar a importância de se analisar a centralidade dos caminhos mínimos. Para isto, foi proposto o uso do algoritmo de Dijkstra para a busca de caminhos mínimos entre todos os pares de vértices em um grafo e para armazenamento e manipulação de dados, foi proposto o uso de dicionários como estrutura de dados mais comum para armazenar os caminhos mínimos com seus respectivos valores de centralidade. O que motiva este trabalho é o fato de que os grafos representam muito bem situações da vida real, como mapas, redes sociais, entre outras. A busca por rotas otimizadas em mapas é um exemplo do escopo de grafos que cobre o problema de caminhos mínimos. É muito interessante analisar os valores de centralidades dos caminhos mínimos em um grafo e, um exemplo disto, no tráfego de dados em uma rede cabeada, existem pontos (rotas) nesta infraestrutura que devem ter estruturas físicas mais robustas para suportar a alta transferência de dados e afins. Pesquisar a centralidade dos caminhos mínimos é uma excelente forma de analisar estes estudos de casos. Os resultados apresentados neste trabalho permitirão visualizar a teoria algorítmica proposta, juntamente com o comportamento da distribuição dos valores de centralidade.

Palavras-chave: Teoria dos grafos. Caminhos mínimos. Algoritmo de Dijkstra.

ABSTRACT

The objective of this work is, through three types of input graphs, to emphasize the importance of the analyses of the centrality of the shortest paths. For this, it was proposed to use the Dijkstra algorithm to search for the shortest paths between all pairs of vertices in a graph and for data store and manipulation, it was proposed to use dictionaries as the most common data structure to store all the shortest paths and their respective values of centrality. What motivates this work is the fact that the graphs represent very well real-life situations, such as maps, social networks, among others. The search for optimized routes on maps is an example of the scope of graphs that covers the shortest path problem. It is very interesting to analyze the values of the shortest paths centralities in a graph and, as an example of this, in the traffic of data in a wired network, there are points (routes) in this infrastructure that these must have more robust physical structures to hold on the high data transfer and alike. Research the centrality of the shortest paths is an excellent way to analyze these case studies. The results presented in this work will allow us to visualize the proposed algorithmic theory, together with the behavior of the distribution of centrality values.

Keywords: Graph theory. Shortest path. Dijkstra's algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
APSP	All-pairs Shortest Path

LISTA DE SÍMBOLOS

$d(v)$	Grau do vértice v
π	pi

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

As redes estão em todo lugar, desde a internet com a estrutura da World Wide Web, economia, geografia à transmissão de doenças, estes tipos de dados são comumente dispostos em grafos, pois são considerados um método muito conveniente para ilustrar visualmente as relações nos dados. O objetivo de um grafo é apresentar dados que são muito numerosos ou complicados para serem descritos de forma adequada no texto e em menos espaço.

Tendo-se que os grafos têm alta representatividade de dados da vida real, a manipulação e análise destes, tal como, a exploração de caminhos mínimos, tópico de discussão deste trabalho, resultam em inúmeras soluções e facilidades para aplicações na realidade. O problema de caminhos mínimos é encontrar um percurso entre dois vértices (ou nós) em um grafo de forma que a soma dos pesos de suas arestas constituintes seja minimizada, este conceito facilmente representa algumas situações reais como busca por rotas otimizadas em mapas, análise de mantenabilidade de rodovias centrais (rotas mais usadas), estudo de transmissibilidade de surtos de doenças (COVID-19), dentre outros. Dado estes exemplos, a busca pela centralidade dos caminhos mínimos nos permite analisar e definir quais são os meios mais comuns de propagação. No exemplo de rodovias, quais precisarão de maior monitoramento para manutenção, no caso de transmissibilidade viral, tentar mitigar os meios mais comuns de transmissão do vírus (rotas aéreas para regiões com surto da doença, por exemplo).

1.2 OBJETIVOS E MOTIVAÇÃO

Este trabalho instiga o estudo e importância da centralidade de caminhos mínimos em um grafo. A análise de centralidade de vértices em grafos tem sido muito discutida e já possui alguns bons algoritmos teóricos e otimizados para isto. A computação da centralidade de caminhos mínimos é um direcionamento, deste mesmo escopo, que ainda carece de algoritmos de viável aplicação na vida real, pois é um processo extremamente custoso, seguindo-se da teoria direta para o cálculo do valor da centralidade. Dito isto, este trabalho irá passar o conceito da metodologia para computação da centralidade dos caminhos mínimos (algoritmo de força bruta), apresenta resultados obtidos, dados numéricos e gráficos visuais, para estudo de comportamento e ressalta pontos adicionais de exploração, como a aleatorização de rotulagem.

1.3 ORGANIZAÇÃO DO TRABALHO

De forma geral, o trabalho está dividido em 3 grandes partes: introdução dos conceitos básicos; apresentação da metodologia para a computação dos caminhos mínimos; apresentação

dos resultados e suas respectivas análises e conclusões. A "introdução"visa definir os conceitos principais para o entendimento do algoritmo proposto neste trabalho, para a computação dos valores de centralidades dos caminhos mínimos, definições como, vértices, arestas, direcionamento, árvores, caminhos, dentre outros. O capítulo "metodologia"apresenta o processo passo a passo do algoritmo para o cálculo dos valores de centralidade dos caminhos mínimos, desde o formato de recepção de dados (dados de entrada)à saída (impressão na saída padrão) dos valores computados. E, por fim, a última parte, "resultados"e "conclusões", apresenta os dados obtidos pelo algoritmo, como as tabelas de centralidade, representação gráfica destes valores, análise de comportamento, juntamente com as conclusões.

2 CONCEITOS INTRODUTÓRIOS

Este capítulo introduz, brevemente, os conceitos básicos de grafos que serão utilizados ao longo dessa dissertação. Os conceitos apresentados neste capítulo podem ser acessados nas seguintes bibliografias: (Bondy e Murty, 1976); (West, 2002); (Bondy e Murty, 2008); (Feofiloff et al., 2018).

2.1 TEORIA DE GRAFOS

Muitas situações do mundo real podem ser convenientemente descritas por meio de um diagrama que consiste em um conjunto de pontos, juntamente com linhas que unem certos pares destes pontos (Bondy e Murty, 1976). Por exemplo, os pontos podem representar pessoas, com linhas unindo pares de amigos ou os pontos podem ser centros de comunicação, com linhas representando links de comunicação. Tais diagramas buscam saber se dois pontos dados são unidos por uma linha, a maneira pela qual eles são unidos é irrelevante. Uma abstração matemática de situações desse tipo dá origem ao conceito de grafos (Bondy e Murty, 2008).

Definição 1 *Um grafo G é denotado por um par ordenado $(V(G), E(G))$, onde $V(G)$ é um conjunto de vértices e $E(G)$ é um conjunto de ligações entre pares de vértices de G, chamados de arestas. São denotadas por $n = |V(G)|$ e $m = |E(G)|$ as cardinalidades dos conjuntos $V(G)$ e $E(G)$, respectivamente.*

2.2 REPRESENTAÇÃO VISUAL

Os grafos têm esse nome porque podem ser representados graficamente. Cada vértice é indicado por um ponto e cada aresta por uma linha que une os pontos que representam suas extremidades (Bondy e Murty, 1976).

Existem diversas maneiras de representação para um grafo. Os pontos (vértices) e conexões (arestas) são meramente um modelo visual da estrutura de dados, isto é, não têm significado.

A seguir, dois exemplos de representações de dois grafos, G e H.

Exemplo 1 *Considere o grafo G, Figura 2.1(a), onde*

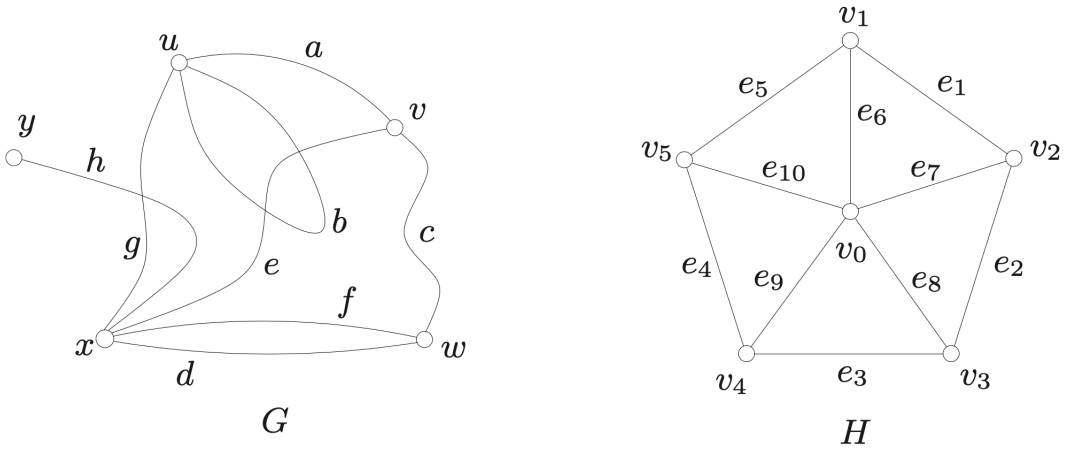
$$V(G) = u, v, w, x, y$$

$$E(G) = a, b, c, d, e, f, g, h$$

Exemplo 2 *Considere o grafo H, Figura 2.1(b), onde*

$$V(H) = v_0, v_1, v_2, v_3, v_4, v_5$$

$$E(H) = e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_10$$



(a) G. (b) H.

Figura 2.1: Representação visual de grafos.

2.3 GRAU DE VÉRTICES E NÚMERO DE ARESTAS

O número de arestas incidentes a um vértice introduz o conceito de grau de um vértice. Algumas bibliografias ainda subdividem este conceito em definições de leques de entrada e saída, ou seja, o grau do leque de saída de um vértice se dá pelas arestas que "saem do vértice"e, de forma intuitiva, o grau de um vértice de entrada é o número de arestas que incidem em um vértice, "entram no vértice". De forma complementar, percebe-se que a soma dos graus de saída de todos os vértices de um grafo é igual ao número de arestas do grafo. A soma dos graus de entrada de todos os vértices também é igual ao número de arestas. Segue daí que um grafo com V vértices tem no máximo $V(V-1)$ arestas. Esse número é apenas um pouco menor que V^2.

Definição 2 *Dado um grafo $G = (V(G), E(G))$, o grau de um vértice v, denotado por $d(v)$, é o número total de arestas de G, incidentes a v.*

2.4 DIRECIONAMENTO

Usando-se das definições anteriores, nesta seção iremos explanar umas das principais propriedades da conectividade de grafos (relacionamento entre vértices de um grafo G). Os grafos podem ou não ser direcionados, pondo-se em exemplos reais, a relação de amizade em um rede social por exemplo, pode ser denotada como não direcionada, pois um amigo a é amigo de b, assim como o inverso. Já em rotas e trajetos (mapas), as ruas possuem direcionamento, isto é, para um exemplo em que uma rua possui mão única, a relação do ponto de interseção de seu início e fim pode ser dado como de a para b, porém não o inverso, pois seria considerado contramão.

Considere a Figura 2.2. Este é um exemplos de um grafo G, não direcionado, onde $V(G) = \{a, b, c, d\}$ e $E(G) = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$. Nota-se que as

arestas $\{a, b\}$ e $\{b, a\}$ são considerados redundantes, motivo pela qual o conjunto $E(G)$ apresenta somente uma delas.

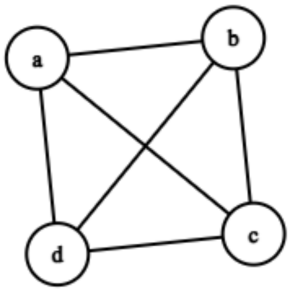


Figura 2.2: Grafo não direcionado.

Considere a Figura 2.3. Este é um exemplos de um grafo G, direcionado, onde $V(G) = \{a, b, c, d\}$ e $E(G) = \{ab, ad, bc, ca, db\}$. A aresta ab, indica que há uma relação de a para b e o inverso não é verdade.

2.5 SUBGRAFOS E SUPERGRAFOS

Dado um grafo G, um subgrafo de G é um grafo com todos os seus pontos e retas em G. Se G', é um subgrafo de G, então G é um supergrafo de G'. Um subgrafo de extensão é um subgrafo, contendo todos os pontos de G. Para qualquer conjunto S de pontos de G, o subgrafo induzido $< S >$ é o subgrafo máximo de G com conjunto de pontos S. Desta forma, dois pontos de S são adjacentes em $< S >$, se e somente se, eles são adjacentes em G.

Na Figura 2.4, G'' é um subgrafo de extensão de G, mas G' não é; G', é um subgrafo induzido, mas G'' não é (Harary, 1994).

Definição 3 *Um grafo H é um subgrafo de G, escrito como $H \subseteq G$ se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.*

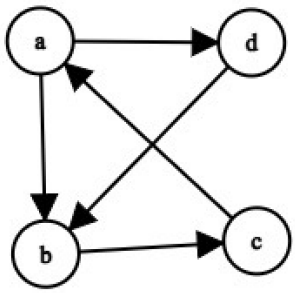
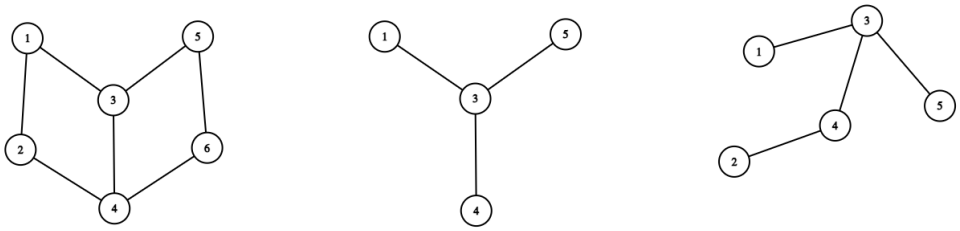


Figura 2.3: Grafo direcionado.



(a) G. (b) G'. (c) G''.

Figura 2.4: Um grafo e dois subgrafos.

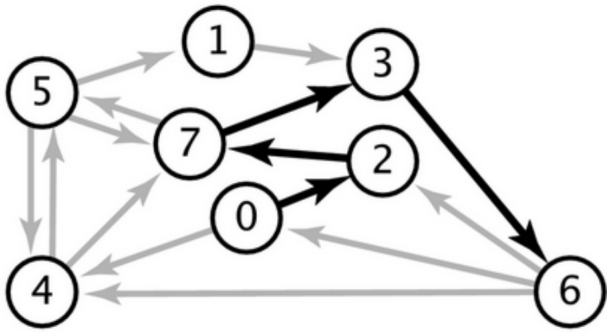


Figura 2.5: Um grafo direcionado com 8 vértices.

2.6 PASSEIOS (*WALK*), CAMINHOS (*PATH*) E CICLOS

Previamente, foram definidos alguns princípios básicos dos grafos, vértices e arestas. Considerando os princípios até agora apresentados, esta seção visa definir as noções de passeios e caminhos em um grafo.

Um passeio (*walk*) em um grafo é uma sequência de vértices dotada da seguinte propriedade: se v e w são vértices consecutivos na sequência, então $v - w$ é uma aresta do grafo (note que o inverso de um passeio não é, em geral, um passeio). Uma aresta do passeio é qualquer aresta $v - w$ do grafo tal que w é o sucessor de v no passeio. Um passeio é fechado (*closed*) se tem pelo menos duas arestas e seu primeiro vértice coincide com o último.

Um caminho (*path*) em um grafo é um passeio sem arestas repetidas, ou seja, um passeio em que as arestas são todas diferentes entre si. Um caminho é *simples* se não tem vértices repetidos. Por exemplo, $0 - 2 - 7 - 3 - 6$ é um caminho simples no grafo G da Figura 2.5.

Todas as arestas de um caminho apontam na mesma direção de um vértice para o seu sucessor. Há quem goste de enfatizar esse fato dizendo caminho dirigido em vez de caminho.

A origem de um caminho é o seu primeiro vértice. O término é o seu último vértice. Se um caminho tem origem s e término t, dizemos que vai de s a t. O comprimento de um caminho

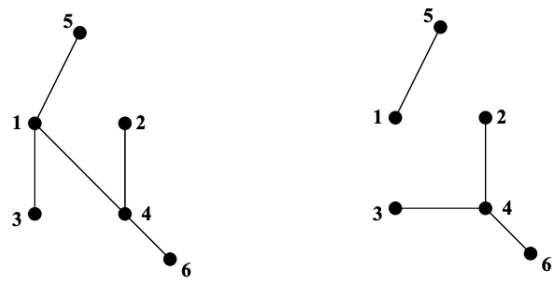


Figura 2.6: Um grafo e uma floresta com duas árvores.

é o número de arestas do caminho. Se um caminho tem n vértices, seu comprimento é pelo menos $n - 1$; se o caminho é simples, seu comprimento é exatamente $n - 1$ (Feofiloff et al., 2018).

Tendo-se das definições acima, um ciclo em um grafo nada mais é que um caminho fechado, ou seja, tem comprimento maior que 1, sem arestas repetidas.

Definição 4 *Dado um grafo $G = (V(G), E(G))$, um passeio $P = (v_1, ..., v_k)$ em G é uma sequência de vértices de G, tal que, para todo par consecutivo de vértices $(v_i - 1, v_i)$, com $2 \leq i \leq k$, temos que $v_i - 1 v_i$ é uma aresta de G. Se não houverem vértices repetidos em P, dizemos que P é um caminho. Um ciclo no grafo G é uma sequência de vértices $C = (v_1, v_2, ..., v_k)$ se $P = (v_1, ..., v_k)$ é um caminho em G e $v_1 v_k \in E(G)$.*

2.7 ÁRVORES E FLORESTAS

Em grafos, uma árvore é um grafo conexo sem ciclos. Uma floresta é um grafo com cada componente conectado em uma árvore. Uma folha em uma árvore é qualquer vértice de grau 1. A Figura 2.6 apresenta uma árvore e uma floresta com duas árvores.

3 CAMINHOS MÍNIMOS E O VALOR DE CENTRALIDADE

Usando-se das propriedades de grafos até agora citados, caminhos mínimos é mais uma propriedade que visa, dado dois vértices v_1 e v_2, buscar qual o caminho mais curto de v_1 para v_2. O *peso* (*ponderação*) de arestas é mais uma propriedade adicional para este capítulo. As arestas de um grafo, além de todas as definições previamente definidas, podem ou não possuir peso, isto é, em um exemplo da vida real, dado um mapa, os pesos das arestas para um grafo que o representasse, poderiam se referir a distância em quilômetros, com isto, o grafo seria classificado como direcionado e ponderado. Quando não se é destacado que o grafo possui pesos, consideramos que todas as suas arestas possuem o mesmo peso ou apenas que não o possuem.

3.1 ALGORITMO DE DIJKSTRA

O algoritmo de Dijkstra resolve o problema de caminhos mínimos de fonte única em um grafo direcionado e ponderado $G = (V, E)$ para o caso em que todos os pesos das arestas não são negativos.

O algoritmo de Dijkstra mantém um conjunto S de vértices cujos pesos finais de caminhos mínimos dos s de origem já foram determinados. O algoritmo seleciona repetidamente o vértice $u \in V - S$ com a estimativa do caminho mais curto mínimo, adiciona-se u a S e relaxa todas as arestas, deixando u. Na implementação a seguir, usa-se uma fila de prioridade mínima Q de vértices, codificados por seus valores de d (Cormen et al., 2009).

```
1       DIJKSTRA(G, w, s)
2       INITIALIZE-SINGLE-SOURCE(G, s)
3       S = \emptyset
4       Q = G * V
5       while Q \neq \emptyset
6             u = EXTRACT-MIN(Q)
7             S = S \cup {u}
8             for each vertex v \in G.Adj[u]
9                   RELAX(u, v, w)
```

O algoritmo de Dijkstra relaxa as bordas, conforme mostrado na Figura 3.1. A linha 2 inicializa os valores de d de maneira usual, e a linha 3 inicializa o conjunto S com o conjunto vazio. O algoritmo mantém a invariante $Q = V - S$ no início de cada iteração do loop $while$ das linhas 5 – 9. A linha 4 inicializa a fila de prioridade mínima Q para conter todos os vértices em V; desde que $S = \emptyset$, naquele momento, o invariante é verdadeiro após a linha 4. Cada vez que, através do loop $while$ das linhas 5 – 9, a linha 6 extrai um vértice u de $Q = V - S$ e a linha 7 o adiciona ao conjunto S, mantendo assim o invariante. (A primeira vez, através deste loop, $u = s$.) O vértice u, portanto, tem a menor estimativa de caminho mínimo de qualquer vértice em $V - S$.

Então, as linhas 8 – 9 relaxam cada aresta (u, v), deixando u, atualizando assim a estimativa $v.d$ e o predecessor $v.\pi$ se pudermos melhorar o caminho mínimo encontrado até agora, passando por u. Observe que o algoritmo nunca insere vértices em Q após a linha 4 e que cada vértice é extraído de Q e adicionado a S exatamente uma vez, de modo que o loop $while$ das linhas 5 – 9 itera exatamente $|V|$ vezes (Cormen et al., 2009).

Como o algoritmo de Dijkstra sempre escolhe o vértice "mais leve"ou "mais próximo"em $V - S$ para adicionar ao conjunto S, dizemos que ele usa uma estratégia "*gulosa*". Este tipo de estratégia nem sempre produzem resultados ótimos (Cormen et al., 2009).

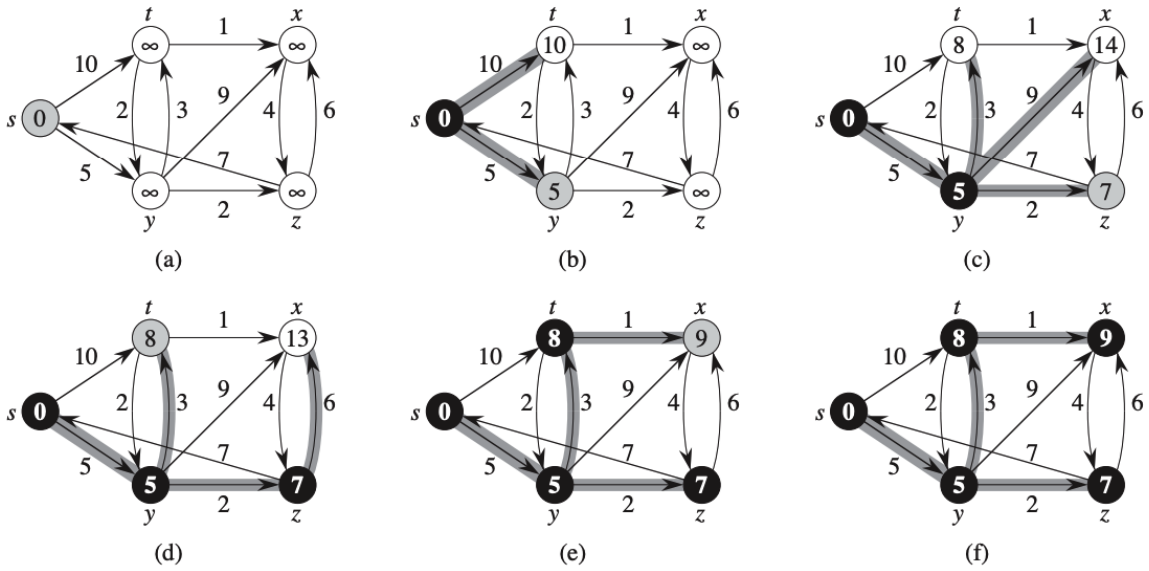


Figura 3.1: A execução do algoritmo de Dijkstra. O s de origem é o vértice mais à esquerda. As estimativas de caminho mais curto aparecem dentro dos vértices e as bordas sombreadas indicam os valores predecessores. Os vértices pretos estão no conjunto S e os vértices brancos estão na fila de prioridade mínima $Q = V - S$. (a) A situação imediatamente antes da primeira iteração do loop $while$ das linhas 5 – 9. O vértice sombreado tem o valor d mínimo e é escolhido como vértice u na linha 6. (b) - (f) A situação após cada iteração sucessiva do laço $while$. O vértice sombreado em cada parte é escolhido como vértice u na linha 6 da próxima iteração. Os valores d e predecessores mostrados na parte (f) são os valores finais (Cormen et al., 2009).

Definição 5 *O algoritmo de Dijkstra, executado em um grafo direcionado ponderado $G = (V, E)$ com função de peso não negativa w e fonte s, termina com $u.d = \delta(s, u)$ para todos os vértices $u \in V$.*

3.2 COMPLEXIDADE DE TEMPO

O tempo de execução no pior caso para o algoritmo de Dijkstra em um grafo com n nós e m arestas é $O(n^2)$ porque permite ciclos direcionados. Ele até encontra os caminhos mais curtos de um nó de origem s para todos os outros nós no grafo. Isso é basicamente $O(n^2)$ para seleção de nó e $O(m)$ para atualizações de distância. Embora $O(n^2)$ seja a melhor complexidade possível para grafos densos, a complexidade pode ser melhorada significativamente para grafos esparsos.

Com pequenas modificações, o algoritmo de Dijkstra pode ser usado como um algoritmo reverso que mantém árvores geradoras mínimas para o nó sorvedouro. Com outras modificações, ele pode ser estendido para se tornar bidirecional. O gargalo no algoritmo de Dijkstra é a seleção de nós. No entanto, usando a implementação de Dial, isso pode ser melhorado significativamente para grafos esparsos (Len et al., 2021).

Encontrar os caminhos mínimos de todos os pares de vértices de um grafo, é uma extensão do problema anterior. É possível resolver este problema, executando um algoritmo de caminhos mínimos de fonte única $|V|$ vezes, uma vez para cada vértice como fonte. Se todos os pesos das arestas forem não negativos, podemos usar o algoritmo de Dijkstra.

3.3 PARTICIONAMENTO DE GRAFOS, UMA INTRODUÇÃO AOS VALORES DE CENTRALIDADE

Para introduzir o conceito de particionamento de um grafo, considere dois exemplos. O primeiro, mostrado na Figura 3.2, descreve as coautorias entre um conjunto de físicos e matemáticos aplicados trabalhando em redes. Pela imagem, percebe-se que existem grupos muito unidos dentro da comunidade, e algumas pessoas que se localizam nos limites de seus respectivos grupos (Easley e Kleinberg, 2010).

Um segundo exemplo, na Figura 3.3, é uma imagem da rede social de um clube de karatê, uma disputa entre o presidente do clube (nó 34) e o instrutor (nó 1), direcionou o clube a se dividir em dois grupos. A Figura 3.3 mostra a estrutura da rede, com os membros nos dois clubes após a divisão indicada pelos nós sombreados e não sombreados.

Os problemas e redes descritos, nos levam a explorar as situações em que o comportamento ou as decisões de uma pessoa dependem das escolhas feitas por outras pessoas, divisão em subgrupos na Figura 3.2 e a influência dos nós 1 e 34 da Figura 3.3 na divisão do clube de karatê em dois subgrupos, seja porque as recompensas da pessoa dependem do que outras pessoas fazem, ou porque as escolhas de outras pessoas transmitem informações que são úteis no processo de tomada de decisão.

3.4 CENTRALIDADE DE INTERMEDIAÇÃO (BETWEENNESS CENTRALITY)

A centralidade de intermediação é uma maneira de detectar a quantidade de influência que um nó tem sobre o fluxo de informações em um grafo. Em outras palavras, usa-se para localizar os nós que servem como "ponte"de uma parte de um grafo para outra ou que inferem possuir maior "importância".

Com o cálculo de todos os pares de caminhos mínimos não ponderados em um grafo, os nós recebem uma pontuação, esta se refere a um montante de "importância"do nó em relação ao número de caminhos mínimos que passam pelo nó. Resumindo, os nós que mais frequentemente estiverem presentes nos caminhos mínimos de um grafo terão pontuações de centralidade de intermediação mais altas.

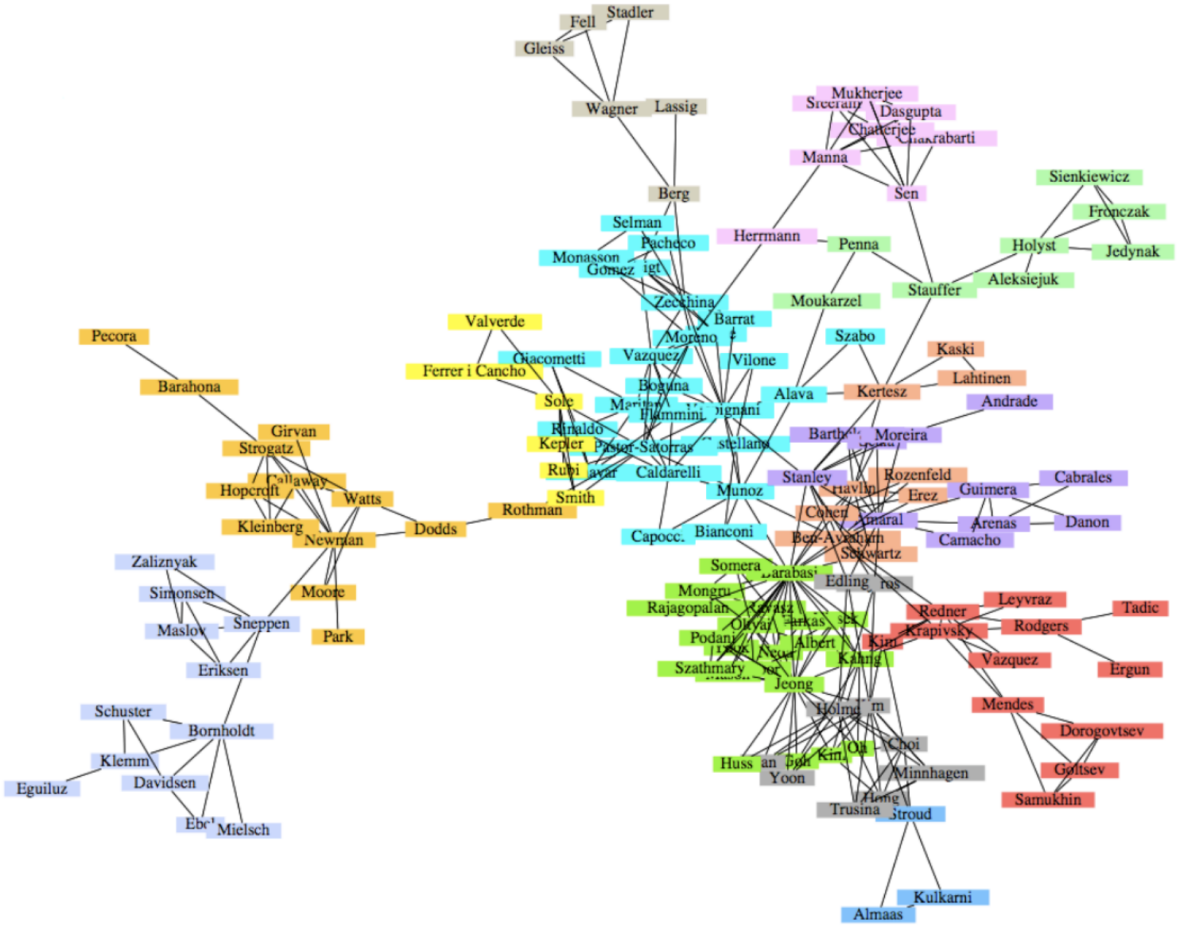


Figura 3.2: Uma rede de coautoria de físicos e matemáticos aplicados.

A Figura 3.4 apresenta um grafo não direcionado colorido com base na centralidade da intermediação de cada vértice do menor (vermelho) ao maior (azul).

Nesta mesma linha, podemos avaliar a "importância"de arestas ou, de forma mais relevante, dos caminhos mínimos, atribuindo-lhes pontuações toda vez que estes estão presentes como sub caminhos mínimos em outros caminhos mínimos. Isto desperta a importância e motivação para estudo de, assim como na centralidade de intermediação de vértices, a centralidade de caminhos mínimos que será discutido na próxima seção.

3.5 CENTRALIDADE DE CAMINHOS MÍNIMOS

Tendo-se discutido a *centralidade de intermediação (betweenness centrality)*, podemos, neste mesmo raciocínio, analisar o problema *All-pairs Shortest Paths (APSP)*, em outras palavras, neste, estamos interessados em calcular os caminhos mínimos com alta centralidade em um grafo.

O *All-pairs Shortest Paths (APSP)* é o problema de calcular o comprimento mínimo entre cada par de vértices em um grafo ponderado. O problema APSP é muito bem estudado e tem havido resultados recentes para uma variedade de suposições para o grafo de entrada (direcionado, não direcionado, ponderado, não ponderado, etc.) (de Lima et al., 2021).

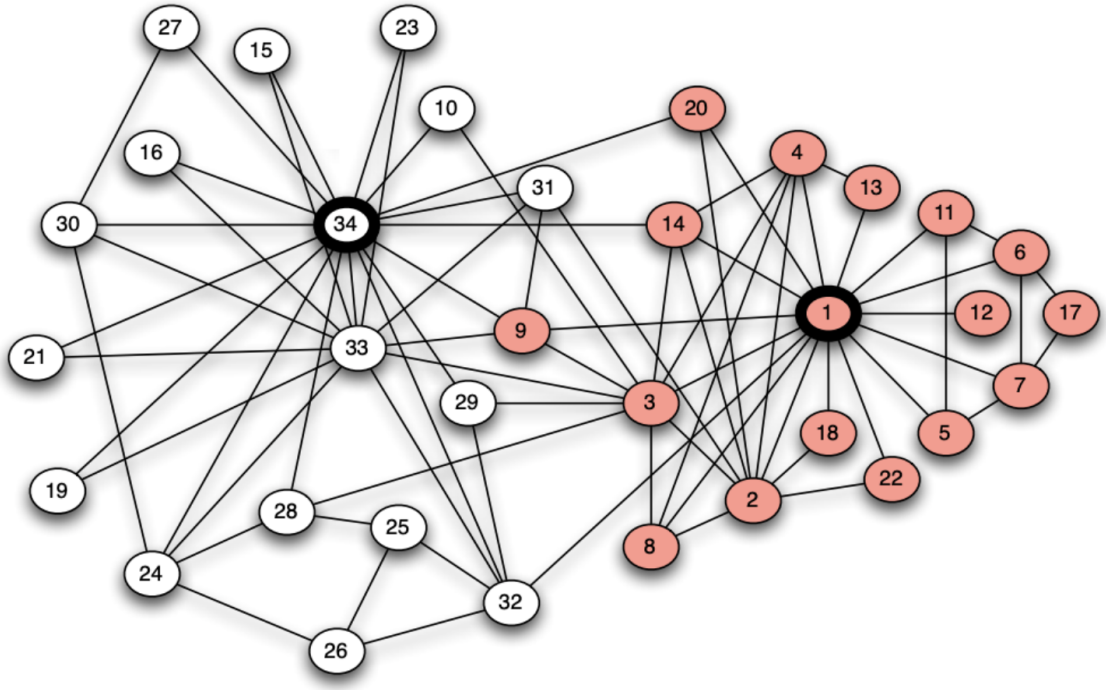


Figura 3.3: Representação de uma rede (grafo) de um clube de karatê.

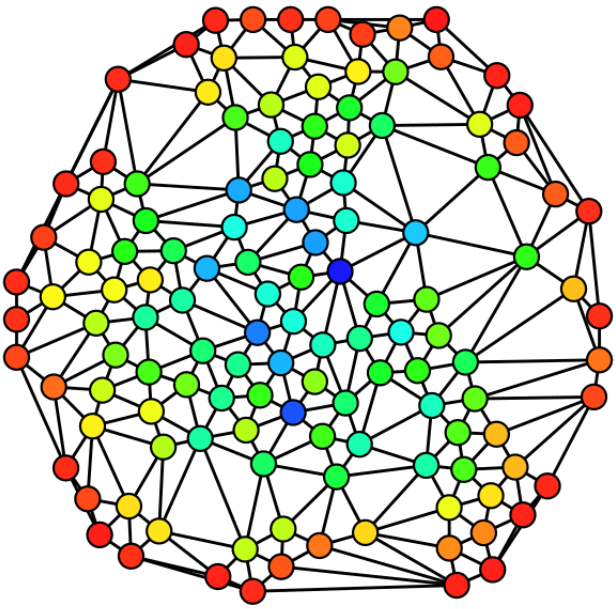


Figura 3.4: Grafo não direcionado com as centralidades de intermediação dos vértices.

Considere o grafo da Figura 3.5, o percurso mais curto do vértice v_1 para o vértice v_3 é o caminho mínimo $<v_1, v_3>$. Calculando-se todos os possíveis pares de caminhos mínimos deste grafo, chegaremos a conclusão de que o caminho mínimo $<v_1, v_3>$ terá valor de centralidade maior que o caminho mínimo $<v_1, v_5, v_4>$ (percurso mais curto do vértice v_1 para o v_4) por exemplo, isto porque as vezes em que o caminho mínimo $<v_1, v_3>$ está presente como subcaminho mínimo é maior que as vezes em que o caminho mínimo $<v_1, v_5, v_4>$ está presente em outros.

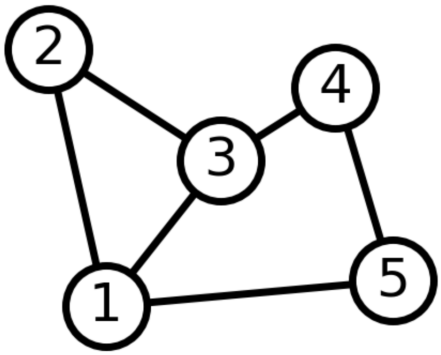


Figura 3.5: Grafo simples com 5 nós.

Adicionalmente, ainda considerando o grafo da Figura 3.5, do vértice v_1 ao vértice v_4, um algoritmo de busca por caminhos mínimos pode retornar o percurso mais curto como sendo $<v_1, v_3, v_4>$ ou $<v_1, v_5, v_4>$, porém nos importa apenas uma destas saídas e usaremos a que o algoritmo em questão encontrar. Para a busca de caminhos mínimos, usaremos o algoritmo de Dijkstra e assumiremos seu retorno como verdade. Esta informação é importante e curiosa, pois dependendo de como o algoritmo se comporta/ trata os dados, o resultado do conjunto de caminhos mínimos podem diferir e, consequentemente, os valores de centralidade dos caminhos mínimos nos grafos que possuam mais de uma opção de percurso mais curso de entre um par de vértices. Este aspecto de aleatorização dos rótulos de um grafo é, inclusive, uma linha de pesquisa neste mesmo escopo, mas algumas bibliografias já indicam que, em grafos muito esparsos, a aleatorização dos rótulos não resulta em diferenças significativas nos valores de centralidade dos caminhos mínimos.

4 METODOLOGIA

De um ponto de vista algorítmico e de estruturas de dados, os grafos têm alta representatividade de redes em geral, sejam estas representações de mapas, redes sociais ou até mesmo o fluxo de influências numa rede bibliográfica. Com isto, existem muitas e das mais diversas linhas de pesquisas sobre os grafos. Os estudos vão desde análise de uma rota otimizada em um mapa à análise do surto de uma transmissão viral. Seguindo-se nesta linha, o estudo sobre a análise da centralidade de caminhos mínimos é muito útil, pois infere quanto à sua importância/ relevância, além de representar diversos cenários da vida real. Um exemplo, dado um mapa, existem rotas que sempre são mais comumente usadas por transportes de carga rodoviário e afins, desta forma, espera-se que este trajeto demande de mais frequente manutenção. Este e muitos outros cenários nos levam a entender a importância do estudo e análise quanto a centralidade de caminhos mínimos.

O problema de centralidades em grafos é um problema muito explorado e já existem diversas técnicas para análise da centralidade de vértices, porém a centralidade de arestas, caminhos mínimos, presente neste mesmo escopo pesquisa, ainda carece de estudos. É um problema extremamente útil de se explorar, pois representa um enorme leque de aplicações da vida real.

Este trabalho é a parte conceitual e de aplicação direta do algoritmo e fluxo teórico proposto no trabalho da Alane de Lima da Universidade Federal do Paraná (de Lima et al., 2021). O problema da centralidade de caminhos mínimos é explorado no trabalho da Alane de um ponto de vista de aplicabilidade, pois o algoritmo e todo seu processo, já em teoria, são extremamente custosos em tempo de execução para grafos muito esparsos. A pesquisa dela busca métodos de análise da centralidade de caminhos mínimos de forma mais eficiente.

Dito isto, este trabalho irá mostrar a aplicação e fluxo de execução da proposta da proposta da Alane, mensurar a centralidade de caminhos mínimos. Os grafos que serão utilizados neste trabalho não são direcionados.

Para apresentação visual e conceitual da metodologia serão utilizados: um grafo simples, com 5 vértices; o modelo de grafo do clube de karatê de Zachary, que é uma rede social de um clube universitário de karatê (Zachary, 2009); a rede de jogos de futebol americano entre as faculdades da Divisão IA durante a temporada regular do outono de 2000 (NetworkX, 2021).

4.1 ALGORITMO

Dados os grafos de entrada, a ideia geral do trabalho é obter todos os possíveis caminhos mínimos, isto é, executar o algoritmo de Dijkstra para cada um dos vértices do grafo. Como já mencionado, o algoritmo de Dijkstra, dado um vértice *target*, constrói uma árvore de caminhos mínimos, sendo cada percurso da raiz até os nós da árvore, um caminho mínimo da raiz ao nó.

Situações em que o grafo é conexo, a quantidade de caminhos mínimos de um vértice para todos os demais é de $n - 1$. Este processo será executado n vezes, isto é, para todos os vértices do grafo.

O trabalho foi desenvolvido em *Python 3* e para desenvolvimento central da aplicação foi usado a biblioteca *NetworkX* (Developers, 2021).

A função *main()*, serve de chamada central de outras subfunções que fazem o tratamento dos dados e que serão explicadas em sequência. De forma geral, esta função faz a construção de um grafo (*linhas 2 - 4*), determina os caminhos mínimos para todos os vértices (*linhas 5 - 8*), faz a computação de centralidade dos caminhos mínimos de um grafo (*linha 7*) e exibe, na saída padrão, todos os caminhos mínimos com seus respectivos valores de centralidade (*linha 8*).

Listing 4.1: Função central para chamada de subfunções.

```
1       def main() -> None:
2             G = simple_graph_generator()
3             G = karate_club_graph_generator()
4             G = football_graph_generator()
5
6             dijkstraTrees = get_dijkstra_trees_from_a_graph(G)
7             shortestPaths = get_shortest_paths(dijkstraTrees)
8             all_shortest_paths_centrality(shortestPaths)
9             print_all_paths_and_centrality(shortestPaths)
```

Para a geração de grafos de entrada, meu código possui 2 funções: o Algoritmo 4.2 (*simple _ graph _ generator()*) e o Algoritmo 4.3 (*karate _ club _ graph _ generator()*). Estas funções usam a biblioteca *NetworkX* para construção manual de um grafo simples (Algoritmo 4.2) e o outro chama um grafo predefinido na própria biblioteca (Algoritmo 4.3). A Figura 4.1 é uma representação visual do grafo gerado pela função *simple _ graph _ generator()*.

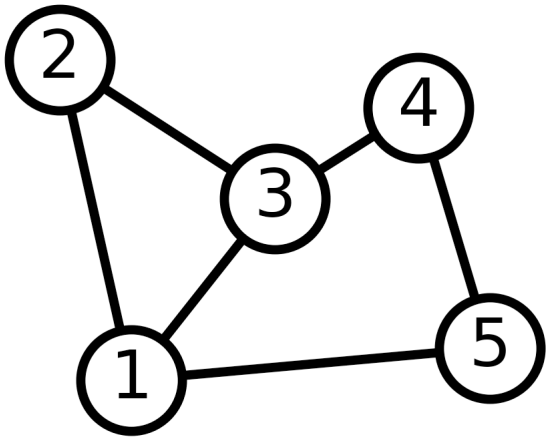


Figura 4.1: Grafo conexo e não direcionado de 5 vértices.

Listing 4.2: Função para gerar um grafo simples.

```
1       def simple_graph_generator():
2             G = nx.Graph()
```

```
            G.add_edge(1, 2)
            G.add_edge(1, 3)
            G.add_edge(1, 5)
            G.add_edge(2, 3)
            G.add_edge(3, 4)
            G.add_edge(4, 5)
            return(G)
```

Listing 4.3: Função para gerar o modelo de grafo do clube de karatê de Zachary.

```
      def karate_club_graph_generator():
            G = nx.karate_club_graph()
            for v in G:
                  print(f"{v:4} {G.degree(v):6}")
            return(G)
```

Tendo os grafos de entrada criados, agora iniciaremos o processo de tratamento destes dados.

Dado um grafo de entrada, o Algoritmo 4.4 (*get _ dijkstra _ trees _from _ a _ graph()*) trata os dados de entrada para passar por cada um dos vértices, gerando a árvore de Dijkstra. Estes dados são armazenados em um dicionário de caminhos mínimos, indexado pelos rótulos dos vértices.

Listing 4.4: Função para tratar os dados de um grafo de entrada e gerar um dicionário de árvores de Dijkstra, indexado pelos rótulos dos vértices do grafo.

```
      def get_dijkstra_trees_from_a_graph(g: dict) -> dict:
            graphDict = {}
            for node in g:
                  if node not in graphDict:
                        graphDict[node] = single_source_dijkstra(g, node)[1]
            return graphDict
```

No próximo passo, o Algoritmo 4.5 (*get _ shortest _ paths _from _ dijkstra _ trees()*) é uma etapa de refinamento dos dados previamente obtidos, isto é, ele pega os dados do dicionário de caminhos mínimos (*get _ dijkstra _ trees _from _ a _ graph()*) e faz algumas operações, como retirada de colchetes nos nomes dos caminhos mínimos e afins.

Listing 4.5: Função para refinamento do dicionário de caminhos mínimos.

```
      def get_shortest_paths_from_dijkstra_trees(graphDict: dict) -> dict:
            pathDict = {}

            for i in graphDict.values():
                  for j in i.values():
                        path = str(j).replace("[", "").replace("]", "")
                        if path not in pathDict and len(path) > 1:
                              newPathInfo = PathInfo(0, len(j))
                              pathDict[path] = newPathInfo
```

```
10
11              return pathDict
```

Agora sim, tendo-se os dados refinados, esta etapa visa calcular a centralidades dos caminhos mínimos, Algoritmo 4.6 (*all _ shortest _ paths _ centrality()*). A centralidades dos caminhos mínimos do grafo é calculada da seguinte forma: para cada caminho mínimo cm_1, verifique em todos os demais caminhos mínimos se cm_1 está presente como subcaminho mínimo (*linhas 2 - 5*); em caso positivo, contabilize +1 na centralidade de cm_1 (*linha 6*); estas etapas são repetidas para todos os demais caminhos mínimos (*linhas 2 - 6*); feito isto, agora a última etapa é referente a normalização dos valores (*linhas 11 - 13*), pois é comumente visto a ponderação de valores com numerações entre 0 e 1, ou seja, quanto mais próximo de 1, maior a centralidade do caminho mínimo.

Listing 4.6: Função para cálculo da centralidade dos caminhos mínimos.

```
1  def all_shortest_paths_centrality(pathDict: dict) -> dict:
2        # Check if i is contained in j. If it is, plus one
3        for i in pathDict:
4              for j in pathDict:
5                    if i in j:
6                          pathDict[i].centrality += 1
7
8        # Get the number of Dijkstra trees
9        numTrees = len(pathDict)
10
11       # Normalizing centrality values
12       for path in pathDict:
13             pathDict[path].centrality /= numTrees * pathDict[path].numNodes
```

5 RESULTADOS

Para a execução dos testes, foram selecionados três grafos, dois representando modelos da vida real, clube de karatê de Zachary e a rede de jogos de futebol americano. Buscou-se selecionar redes que contivessem número de vértices e arestas distintas, para que fosse possível obter resultados não viciados em relação ao tamanho do grafo sobre os quais seriam executados os testes. Dito isso, as redes selecionadas representam diferentes tipos de redes e possuem quantidades de vértices que variam de 5 a 115. A Tabela 5.1 apresenta a descrição das redes nas quais foram executados os testes propostos.

Tabela 5.1: Detalhamento das redes selecionadas.

Rede	Vértices	Arestas	Tipo	Classificação
Simple	5	6	Não direcionado	Teste
Zachary	34	78	Não direcionado	Social
Football	115	613	Não direcionado	Informação

A Tabela 5.2 representa os valores de centralidade calculados para o grafo mais simples, Algoritmo 4.2. A ordem apresentada na tabela depende muito do contexto de análise, para este grafo, eu fiz a ordenação de tal forma a agrupar ou deixar mais próximos os vértices iguais, por exemplo, as primeiras linhas desta tabela estão ordenadas primeiro com todos os caminhos mínimos que iniciam no vértice 1 e assim por diante.

Tabela 5.2: Valores de centralidade do grafo simples, Algoritmo 4.2.

Path	Centrality
[1, 2]	[0.05]
[1, 3]	[0.075]
[1, 5]	[0.075]
[1, 3, 4]	[0.016666666666666666]
[2, 1]	[0.05]
[2, 3]	[0.05]
[2, 1, 5]	[0.016666666666666666]
[2, 3, 4]	[0.016666666666666666]
[3, 1]	[0.075]
[3, 2]	[0.05]
[3, 4]	[0.075]
[3, 1, 5]	[0.016666666666666666]
[5, 1]	[0.075]
[5, 4]	[0.025]
[5, 1, 2]	[0.016666666666666666]
[5, 1, 3]	[0.016666666666666666]
[4, 3]	[0.075]
[4, 5]	[0.025]
[4, 3, 1]	[0.016666666666666666]
[4, 3, 2]	[0.016666666666666666]

Já para os grafos do clube de karatê de Zachary e da rede de jogos de futebol americano irei apresentar dois gráficos, representando a distribuição dos valores de centralidade dos

caminhos mínimos, pois, a quantidade de caminhos mínimos cresce exponencialmente de acordo com o número de vértices, desta forma, a tabela para estes dois grafos são muito extensos. O Gráfico 5.1 e o Gráfico 5.2 distribuem os valores de centralidade dos caminhos mínimos no eixo y, de forma ordenada (crescente), enquanto o eixo x representa apenas os caminhos mínimos em si, ou seja, estão rotulados sequencialmente, pois nesta visualização o percurso exato de cada um dos caminhos mínimos não nos interessa. Outra informação importante, devido aos valores de centralidade serem o ponto focal, para a representação gráfica, foram mesclados os valores de centralidade coincidentes.

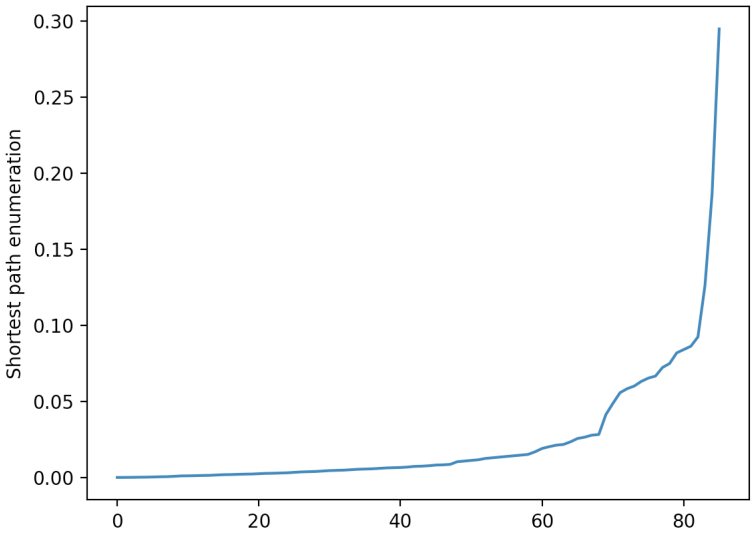


Figura 5.1: Gráfico com a distribuição de valores de centralidade do grafo do clube de karatê de Zachary.

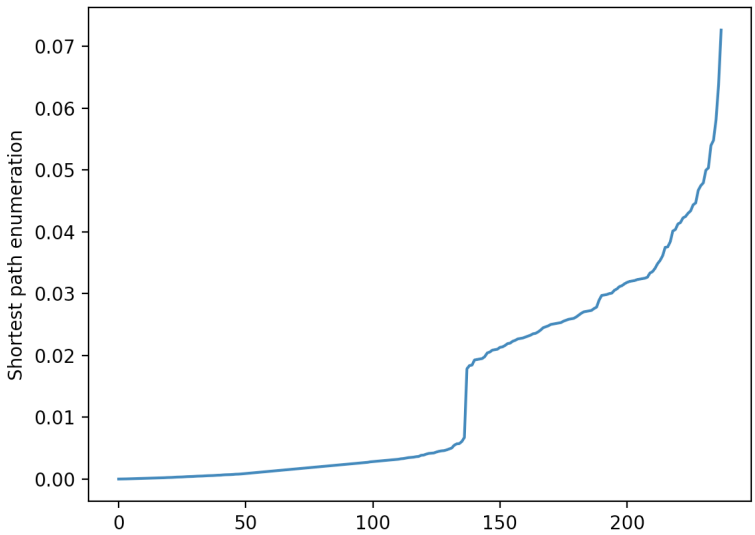


Figura 5.2: Gráfico com a distribuição de valores de centralidade do grafo da rede de jogos de futebol americano.

6 CONCLUSÕES

Neste trabalho, foram usados três tipos de grafos de entrada, um deles é muito simples e não reflete situações da vida real, que são os tipos de aplicações que nos interessam. O propósito do uso de um grafo mais simples neste trabalho é permitir o entendimento mais claro do algoritmo para a computação da centralidade de caminhos mínimos proposto. Já os outros dois grafos, são modelos que refletem condições da vida real, dito isto, podem ser inviáveis ou muito difíceis de analisar visualmente ou por *teste de mesa*, mas é muito interessante e útil ver o comportamento e a distribuição dos valores de centralidade dos caminhos mínimos nestes, pois, pelos gráficos obtidos, constata-se que os grafos possuem um comportamento muito parecido entre eles, segundo a relação de caminhos mínimos.

Vale ressaltar que o algoritmo proposto neste trabalho para o problema de caminhos mínimos é muito custoso. Dito isto, pode se tornar inviável para uso em algumas aplicações da vida real, porém este trabalho visa despertar a curiosidade referente a importância e utilidade deste tipo de análise.

Do tempo de execução, pelos testes realizados é possível constatar que o tempo de processamento do grafo de entrada e cálculo dos valores de centralidade para todos os caminhos mínimos, cresce exponencialmente conforme o aumento do número de vértices. Importante destacar isto, pois corrobora a necessidade de se explorar algoritmos mais eficientes, inclusive, já existem propostas de algoritmos que calculam o valor da centralidade de caminhos mínimos por meio de amostragem (de Lima et al., 2021).

Da análise dos valores de centralidade dos caminhos mínimos. De forma intuitiva, já se percebe um comportamento e relação entre o tamanho do percurso de um caminho mínimo e seu valor de centralidade, em outras palavras, caminhos mínimos com um percurso maior tendem a ter valores de centralidade menores. Explicando, dado um caminho mínimo, ele só poderá estar contido como subcaminho em algum outro, se e somente se, este conter pelo menos a mesma quantidade de vértices, ou seja, quanto menor o percurso, mais possibilidades de este estar contido como subcaminho mínimo em outros e, consequentemente, maior o seu valor de centralidade. Em resumo, pode-se dizer que quanto maior o percurso de um caminho mínimo, menor seu valores de centralidade e, de forma análoga, quanto menor o percurso de um caminho mínimo, mais provável de seu valor de centralidade ser maior.

Do comportamento do gráfico de distribuição dos valores de centralidade dos caminhos mínimos. O Gráfico 5.1 e o Gráfico 5.2 apresentam a disposição dos valores de centralidade ordenados de forma crescente, isto nos permite constatar que a curva é sempre crescente. Isto é importante destacar, pois indica que os caminhos mínimos tendem a ser muito centrais ("importantes") ou pouco centrais e, outro fator interessante disto, assim como supracitado, os caminhos mínimos dispostos mais à esquerda, os valores menores, referem-se aos caminhos

mínimos com percurso maior, enquanto que, de forma análoga, os caminhos mínimos dispostos mais à direita dos gráficos, representam os percursos maiores.

Dito tudo isto, vale ressaltar um aspecto em adicional. Sabemos que existem diversas situações em que, dado um grafo, existe a possibilidade de ocorrer mais de uma opção de caminho mínimo entre um par de vértices, isto é um aspecto importante, pois, de forma lógica, os valores de centralidade possivelmente mudem. Em grafos pequenos este aspecto pode ser facilmente percebido e por isso é necessário assumir como verdade o retorno de um algoritmo de busca por caminhos mínimos, pois cada algoritmo pode tratar de forma diferente a busca. Existem algumas linhas de pesquisas que indicam que o problema da aleatorização de rotulagem tem impacto desprezível em grafos muito esparsos, entende-se disto que a aleatorização de rotulagem não é, necessariamente, um empecilho para o estudo da centralidade de caminhos mínimos em um grafo (de Lima et al., 2021).

REFERÊNCIAS

Bondy, J. A. e Murty, U. S. R. (1976). *Graph theory with applications*. Elsevier Science Publishing Co., Inc.

Bondy, J. A. e Murty, U. S. R. (2008). *Graph Theory*. Springer London.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. e Stein, C. (2009). *Introduction to Algorithms*. Massachusetts Institute of Technology.

de Lima, A., da Silva, M. e Vignatti, A. (2021). Shortest path centrality and the apsp problem via vc-dimension and rademacher averages. `https://www.inf.ufpr.br/murilo/public/apsp-alg.pdf`. Acessado em 15/11/2021.

Developers, N. (2021). Networkx. `https://networkx.org/documentation/stable/index.html`. Acessado em 26/11/2021.

Easley, D. e Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.

Feofiloff, P., Kohayakawa, Y. e Wakabayashi, Y. (2018). Uma introdução sucinta à teoria dos grafos. `https://www.ime.usp.br/~pf/teoriadosgrafos/`. Acessado em 06/11/2021.

Harary, F. (1994). *Graph Theory*. Addison-Wesley Publishing Company.

Len, G., Andreas, L. e Eric, W. (2021). Dijkstra's algorithm. `https://mathworld.wolfram.com/DijkstrasAlgorithm.html`. Acessado em 15/11/2021.

NetworkX (2021). Football. `https://networkx.org/documentation/stable/auto_examples/graph/plot_football.html`. Acessado em 29/11/2021.

West, D. (2002). *Introduction to Graph Theory*. Pearson Education (Singapore) Pte. Ltd., Indian Braneh, 482 F.I.E. Patparganj, Delhi 1 10 092, India.

Zachary, W. W. (2009). *An Information Flow Model for Conflict and Fission in Small Groups*. University of New Mexico.