LORENA KEROLLEN BOTELHO TAVARES

UM ESTUDO SOBRE O PROBLEMA DOS *K* CAMINHOS MAIS CURTOS

LAVRAS – MG

2021

LORENA KEROLLEN BOTELHO TAVARES

UM ESTUDO SOBRE O PROBLEMA DOS *K* CAMINHOS MAIS CURTOS

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Mayron César de Oliveira Moreira
Orientador

LAVRAS – MG

2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Tavares, Lorena Kerollen Botelho.
Um estudo sobre o problema dos K caminhos mais curtos / Lorena Kerollen Botelho Tavares. – Lavras : UFLA, 2021.
33 p. : il.

Monografia(graduação)–Universidade Federal de Lavras, 2021.
Orientador: Prof. Dr. Mayron César de Oliveira Moreira.
Bibliografia.

1. Problema do K caminhos mais curtos. 2. Algoritmo de Yen. 3. Algoritmos de Desvio. I. Moreira, Mayron César de Oliveira. II. Título.

LORENA KEROLLEN BOTELHO TAVARES

UM ESTUDO SOBRE O PROBLEMA DOS *K* CAMINHOS MAIS CURTOS
A STUDY ABOUT THE *K* SHORTEST PATHS PROBLEM

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADA em 09 de Março de 2021.

Prof. Dr. Mayron César de Oliveira Moreira	UFLA
Prof. Dr. Tiago Januario	UFBA
Prof. Dr. Luiz Henrique Andrade Correia	UFLA

Prof. Dr. Mayron César de Oliveira Moreira
Orientador

LAVRAS – MG
2021

Dedico este trabalho a todos que contribuíram e se fizeram presentes durante seu desenvolvimento.

AGRADECIMENTOS

Agradeço primeiramente à minha família, por sempre me apoiar em minhas decisões, e por terem contribuído enormemente para que eu fosse capaz de finalizar essa etapa da minha vida.

Agradeço a todas as minhas amigas e amigos por terem feito parte deste ciclo que se encerra, e por terem me auxiliado durante toda a trajetória.

Agradeço ao meu orientador Dr. Mayron César de Oliveira Moreira, pela paciência e dedicação durante todo o tempo de orientação.

Agradeço à Universidade Federal de Lavras, em especial ao departamento de Ciência da Computação, por ter contribuído com todo o conhecimento que adquiri durante a graduação.

E por fim, agradeço a todas as pessoas que fizeram parte da minha vida durante esses anos, vocês foram de suma importância para que eu chegasse até aqui.

"All we have to decide is what to do with the time that is given us."
(J. R. R. Tolkien)

RESUMO

O problema dos K caminhos mais curtos acíclicos (*K-SP*, do inglês *K Shortest Loopless Paths Problem*) consiste em encontrar, em uma rede direcionada valorada, um conjunto de K rotas, minimizando o caminho percorrido entre um ponto de origem e um ponto de destino. O objetivo deste trabalho é realizar um estudo sobre um algoritmo clássico da literatura para o *K-SP*, apresentando uma explicação detalhada do método e do conceito de algoritmo de desvio. Espera-se que o trabalho desenvolvido seja um facilitador para o entendimento do tema, tornando seu conteúdo mais compreensível para a comunidade acadêmica com interesse na resolução do problema e no algoritmo descrito.

Palavras-chave: Problema dos K Caminhos Mais Curtos. Algoritmo de Yen. Algoritmos de desvio.

ABSTRACT

The *K* Shortest Loopless Paths Problem *(K-SP)* consists of finding, in a weighted network, a set of *K* routes, minimizing the path taken between an initial point and a terminal point. This work aims to study a classic literature algorithm for *K-SP*, presenting a detailed explanation of the method and the concept of deviation algorithm. We expect that the work developed will help understand the theme, making its content more understandable for academic community interested in the problem resolution and the algorithm described here.

Keywords: *K* Shortest Paths Problem. Yen's Algorithm. Deviation Algorithms.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O problema dos *K* caminhos mais curtos (*K-SP*, do inglês *K Shortest Paths*) foi introduzido a partir do trabalho de (HOFFMAN; PAVLEY, 1959). O *K-SP* diz respeito a encontrar, em uma rede direcionada valorada, os *K* caminhos de menor custo entre um ponto de origem e um ponto de destino. Neste estudo, abordamos a variante do *K-SP* acíclica (conhecida na literatura como *K Shortest Loopless Paths Problem*), que não permite ciclos para o cálculo dos caminhos mais curtos.

O algoritmo de Yen (YEN, 1971), um clássico da literatura do *K-SP*, foi escolhido para análise e implementação.

1.1 Motivação e Justificativa

O *K-SP* está presente em situações reais, em que encontrar um único caminho mais curto entre dois pontos distintos de uma rede não é suficiente. Há diversos fatores que influenciam na escolha de um caminho. Assim sendo, minimizar a distância entre dois pontos quaisquer pode não ser o único fator a ser levado em consideração quando existe a necessidade de obter um caminho. Problemas que possuem restrições adicionais que são mais complexas de otimizar são mencionados por (EPPSTEIN, 1998). Um desses problemas surge no contexto de transmissão de energia, onde é necessário selecionar uma rota de transmissão, mas a comunidade local pode indicar a preferência para a escolha de trechos da rede. Nesse caso, uma possível solução é construir mais de um caminho e escolher de acordo com os critérios analisados individualmente.

Problemas reais são passíveis de variabilidade dos seus parâmetros, devido a incertezas como tempo para percorrer de um ponto a outro, ou até mesmo inviabilidade de deslocamento em um trecho inicialmente viável. Por isso, ter a informação de mais de um caminho de menor custo é relevante na prática.

Para problemas maiores, em termos de dimensionalidade da rede, os autores (VANHOVE; FACK, 2012) tratam do problema considerando instâncias de até 10.000 pontos. Apesar da heurística proposta não encontrar o conjunto exato dos K caminhos mais curtos, ela é executada em um período de tempo menor, e de acordo com os autores, os caminhos encontrados tem desvio de 1% em relação à solução ótima.

Ademais, a escolha do algoritmo de Yen se deu pela sua relevância na literatura. O algoritmo de Yen (YEN, 1971) tem aproximadamente 2.550 citações no Google Scholar (informação de 15 de fevereiro de 2021), e é, ao que consta, o algoritmo mais citado para o *K-SP* acíclico.

1.2 Objetivos

O objetivo do trabalho é realizar um estudo sobre o algoritmo de Yen (YEN, 1971), presente na literatura para solucionar o problema descrito, realizando uma explicação detalhada de como ele encontra os K caminhos mais curtos de uma rede, e o conceito de algoritmo de desvios que ele utiliza para tal.

Espera-se que o trabalho desenvolvido seja um facilitador para o entendimento do tema, tornando seu conteúdo mais compreensível para a comunidade acadêmica com interesse na resolução do problema e nos algoritmos descritos.

1.3 Organização do Trabalho

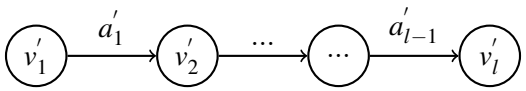
O Capítulo 2 apresenta a definição formal do *K-SP*. No Capítulo 3, são apresentados os conceitos de algoritmos de desvio, utilizado no algoritmo estudado. O Capítulo 4 é destinado às explicações do algoritmo de Yen. O Capítulo 5 mostra uma análise experimental do resultado obtido com a execução do algoritmo. Por fim, o Capítulo 6 conclui o trabalho proposto.

2 PROBLEMA DOS K CAMINHOS MAIS CURTOS

As notações e definições formais apresentadas abaixo são baseadas no trabalho de (MARTINS; PASCOAL, 2003). Dada uma rede (N,A) direcionada, onde N é o conjunto de vértices, com $N = \{v_1, ..., v_n\}$, e A é o conjunto de arcos, com $A = \{a_1, ..., a_m\} \subset N \times N$, o problema dos K caminhos mais curtos acíclicos (*K-SP*) pode ser definido pela extração dos caminhos a partir da rede (N,A). Tanto N quanto A são conjuntos finitos da rede (N,A). Um arco a_k é representado por dois vértices (v_i, v_j), com $v_i \neq v_j$, $v_i, v_j \in N$.

A definição de caminho é feita através de p, $p = \langle v_1^{'}, a_1^{'}, v_2^{'}, ..., a_{l-1}^{'}, v_l^{'} \rangle$ é um caminho de i a j (dois vértices em (N,A)), como demonstrado na Figura 2.1.

Figura 2.1 – Caminho p de 1 a l.



Fonte: criado a partir de (MARTINS; PASCOAL, 2003).

- $v_k^{'} \in N$ para qualquer $k \in \{1, ..., l\}$;
- $a_k^{'} = (v_k^{'}, v_{k+1}^{'}) \in A$ para qualquer $k \in \{1, ..., l-1\}$.

Portando, sabendo que não existem vértices repetidos em um mesmo caminho, em p existem l vértices diferentes entre si, e $l-1$ arcos de i a j.

Considerando a definição acima, temos $i = v_1^{'}$ e $j = v_l^{'}$. Cada vértice sozinho é dito ser um caminho degenerado, sem nenhum arco. Um caminho é dito acíclico quando não possui algum vértice repetido. Por fim, P_{ij} é o conjunto dos caminhos em (N,A) de i (vértice inicial) a j (vértice final).

Para definição de subcaminho, seja x e y dois vértices de um caminho $p \in P_{ij}$. Um caminho $q \in P_{xy}$ é chamado de subcaminho de p se ele coincide com p de x a y. Coincidindo, temos que x e y são dois vértices entre i e j. A representação de subcaminho fica como $q = \text{sub}_p(x,y)$.

Para cada conjunto de caminhos P_{ij}, existe um custo associado real de i à j, que pode ser definido pela Equação (2.1). Para cada $p \in P_{ij}$, o somatório dos custos dos arcos é dado por (2.2).

$$c : \bigcup_{i,j \in N} P_{ij} \longrightarrow \mathbb{R} \tag{2.1}$$

$$p \longrightarrow c(p) = \sum_{(x,y) \in p} c_{xy} \tag{2.2}$$

De acordo com a expressão acima, o custo do caminho p, $c(p)$, é o somatório dos custos dos arcos em p. Para a definição de conjunto de caminhos mais curtos, seja s e t dois vértices diferentes de (N,A), s sendo vértice inicial e t vértice final. Para simplificar, P_{st} será chamado apenas de P.

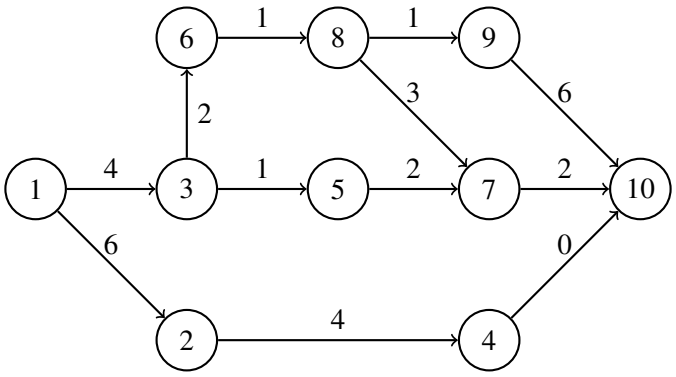
Dado um inteiro positivo K, o problema dos K caminhos mais curtos acíclico consiste em encontrar um conjunto $P_K = \{p_1, ..., p_K\} \subseteq P$ de caminhos mais curtos, de forma que as seguintes restrições sejam satisfeitas:

- p_k não possui ciclos, para qualquer $k \in \{1, ..., K\}$;
- $c(p_k) \leq c(p_{k+1})$, para qualquer $k \in \{1, ..., K-1\}$;
- $c(p_K) \leq c(p)$, para qualquer caminho acíclico $p \in P - P_K$;
- p_k é determinado antes de p_{k+1}, para qualquer $k \in \{1, ..., K-1\}$.

3 ALGORITMOS DE DESVIO

Algoritmos desenvolvidos para resolver o *K-SP* utilizam, em sua maioria, o conceito de algoritmos de desvio. Tais algoritmos utilizam desvios dentro de determinado caminho pré-determinado, a fim de encontrar outras possibilidades de percursos de uma mesma rede. Assim, algoritmos de desvio são utilizados para o ranqueamento de K caminhos mais curtos.

Figura 3.1 – Rede (N,A).

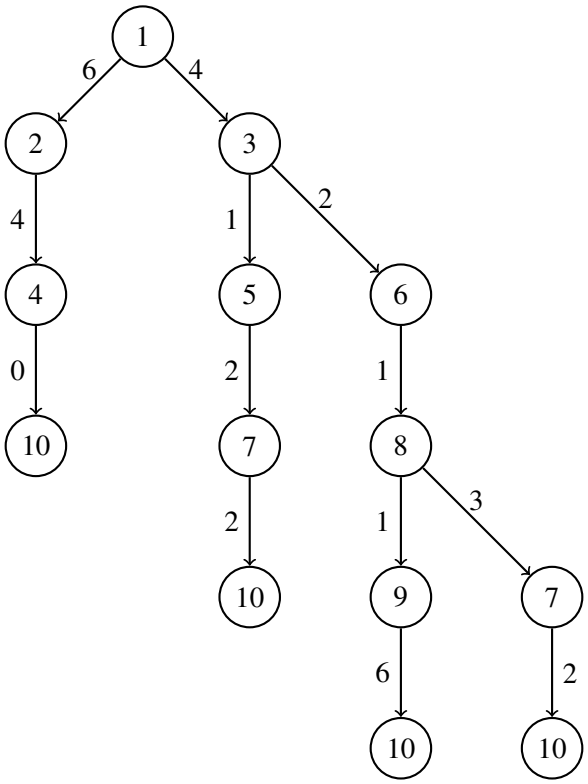


Fonte: elaborado pela autora.

Em (MARTINS; PASCOAL, 2003), os autores apresentam o conceito de algoritmo de desvio como uma pseudo-árvore de caminhos criados com base nos desvios possíveis na rede (N,A). A estrutura é chamada de pseudo-árvore porque pode conter vértices repetidos[1]. No entanto, como o mesmo vértice é diferente em cada caminho, sendo conectado por arestas distintas, o termo pseudo-árvore é utilizado. A partir da rede na Figura 3.1, um exemplo de uma pseudo-árvore de caminhos de desvios para a rede pode ser visto na Figura 3.2.

[1] A definição formal de árvore consiste em um grafo acíclico e conexo.

Figura 3.2 – Pseudo-árvore de caminhos para $K = 4$.



Fonte: elaborado pela autora.

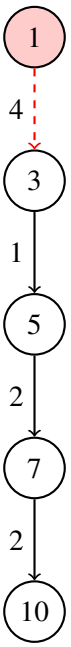
Tabela 3.1 – Definições de símbolos utilizados.

Símbolo	**Definição**
	vértice de desvio
	arco de desvio
	caminho raiz
	caminho derivado

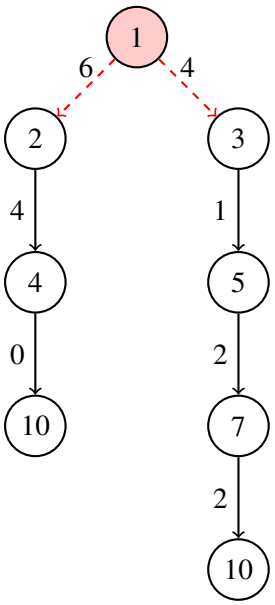
Fonte: adaptado de (CHEN et al., 2020).

Figura 3.3 – Caminhos encontrados na rede (N,A), para $K = 3$.

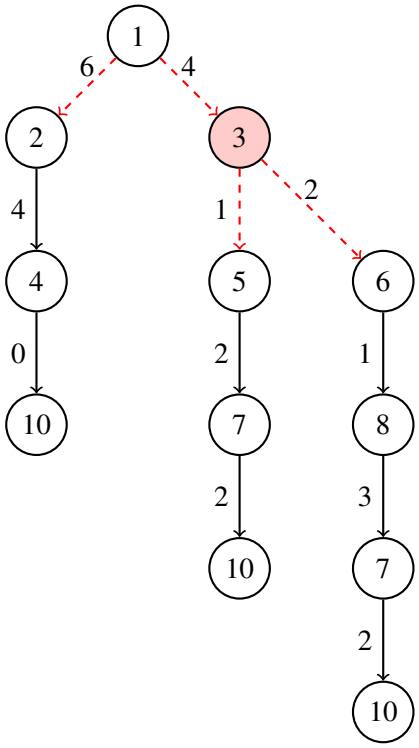
(a) Pseudo-árvore com caminho p_1.



(b) Pseudo-árvore com caminho p_1 e p_2.



(c) Pseudo-árvore com caminho p_1, p_2 e p_3.



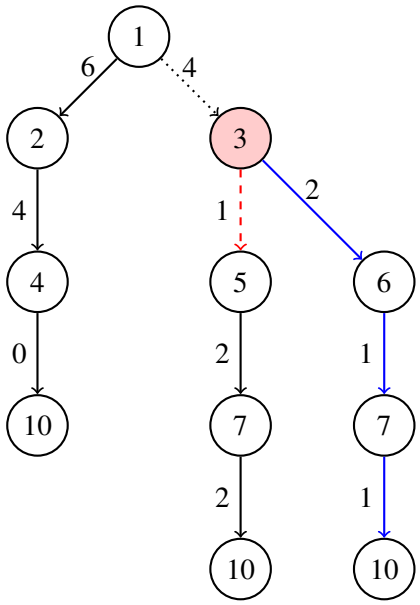
Fonte: elaborado pela autora.

De acordo com (MARTINS; PASCOAL, 2003), o vértice de desvio é o vértice do caminho p_j (p_j sendo o último caminho adicionado a pseudo-árvore de caminhos mais curtos) mais distante do vértice de origem, considerando quantidade de vértices intermediários, e que possua ligação com algum dos $j-1$ caminhos já determinados. O vértice de desvio do caminho mais curto da rede, p_1, é o primeiro vértice. Após mais caminhos serem adicionados a pseudo-árvore de caminhos mais curtos, é necessário verificar qual vértice no caminho p_j satisfaz as condições mencionadas anteriormente.

Para exemplificar, na Figura 3.3, o primeiro vértice de desvio do caminho 3.3a é o vértice v_1, pois há apenas o caminho p_1 na pseudo-árvore. No caminho 3.3b, o vértice v_1 é novamente o vértice de desvio, uma vez que ele é o vértice mais distante da origem que ainda coincide com um vértice de outro caminho da pseudo-árvore. Já para o caminho 3.3c, o primeiro vértice de desvio do caminho p_3 é o vértice v_3, pois ele é o vértice mais distante da origem que ainda coincide com algum dos $j-1$ ($j=3$) caminhos já determinados (nesse caso, coincidindo com p_1).

O arco de desvio é o arco subsequente ao vértice de desvio, sendo retirado do caminho p_j para criar o desvio que gera novos caminhos. Na Figura 3.3a, o arco de desvio é o arco que liga v_1 a v_3. Durante a execução do algoritmo que busca os caminhos de desvios - para este trabalho, o Algoritmo de Yen - o arco de desvio é retirado da rede para que novos caminhos mais curtos sejam encontrados. Como pode ser visto na Figura 3.3b e na Figura 3.3c, mais de um arco é considerado arco de desvio: os arcos (v_1, v_2), (v_1, v_3), e os arcos (v_1, v_2), (v_1, v_3), (v_3, v_5) e (v_3, v_4), respectivamente. Isso acontece para impedir que o algoritmo de Dijkstra - utilizado para encontrar o caminho mais curto a cada iteração - encontre mais de uma vez o mesmo caminho.

Figura 3.4 – Pseudo-árvore para exemplificar caminho raiz e caminho derivado.



Fonte: elaborado pela autora.

O caminho raiz é todo o caminho até o vértice de desvio. Consiste na parte do caminho original que se repete no caminho a ser criado. Na Figura 3.4, o caminho raiz é composto pelo arco (v_1, v_2).

O caminho derivado é o caminho formado a partir do vértice de desvio após retirada do arco de desvio do caminho original. Ele é o caminho que compõe o desvio tomado do vértice de desvio até o vértice de destino. Na Figura 3.4, o caminho derivado aparece como os arcos (v_3, v_6), (v_6, v_7) e (v_7, v_{10}).

A pseudo-árvore de caminhos mais curtos, representada pela Figura 3.2, consiste no conjunto L de caminhos determinados no pseudo-código 1. L é o conjunto resultante com os K caminhos mais curtos de (N,A). O exemplo 3.2 possui $K = 4$ e foi obtido a partir da rede (N,A) em 3.1. Os 4 caminhos mais curtos dessa rede são:

- $p_1 = \langle v_1, (v_1, v_3), v_3, (v_3, v_5), v_5, (v_5, v_7), v_7, (v_7, v_{10}), v_{10} \rangle$, com $c(p_1) = 9$.

- $p_2 = \langle v_1, (v_1, v_2), v_2, (v_2, v_4), v_4, (v_4, v_{10}), v_{10} \rangle$, com $c(p_2) = 10$.
- $p_3 = \langle v_1, (v_1, v_3), v_3, (v_3, v_6), v_6, (v_6, v_8), v_8, (v_8, v_7), v_7, (v_7, v_{10}), v_{10} \rangle$, com $c(p_3) = 12$.
- $p_4 = \langle v_1, (v_1, v_3), v_3, (v_3, v_6), v_6, (v_6, v_8), v_8, (v_8, v_9), v_9, (v_9, v_{10}), v_{10} \rangle$, com $c(p_1) = 14$.

A eficiência da árvore de desvios em encontrar os vértices de desvios para o cálculo dos caminhos derivados diz respeito a como os dados utilizados na escolha desse vértice são obtidos. Armazenando a quantidade de ramificações de cada vértice (considerando seus arcos divergentes), para escolher o primeiro vértice de desvio a cada iteração do algoritmo de Yen, basta percorrer o caminho p_j do último vértice ao primeiro. Assim, o primeiro vértice a ser encontrado que possua mais de uma ramificação pode ser selecionado como vértice de desvio.

O Algoritmo 1 apresenta o pseudo-código de um algoritmo genérico de caminho de desvio. A explicação do pseudo-código é feita seguindo a numeração de suas linhas.

Algorithm 1 Algoritmo genérico de caminho de desvio.

Entrada: Rede (N,A), vértice de origem s, vértice de destino t, quantidade de caminhos mais curtos acíclicos K.

Saída: Conjunto L de caminhos determinados.

```
 1: p_1 := getCaminhoMaisCurto((N,A), s)
 2: C := {p_1}
 3: L := ∅
 4: for j := 1 to K do
 5:     if C.isEmpty() then
 6:         return L
 7:     end if
 8:     p_j := heapPop(C)
 9:     L := add(L, p_j)
10:     D_j := algoritmoDeYen(p_j, L)
11:     C := union(C, D_j)
12: end for
13: return L
```

- **Linha 1:** O algoritmo de Dijkstra (DIJKSTRA, 1959) um para todos é utilizado para obtenção do caminho mais curto da rede (N,A). Assim, a função *getCaminhoMaisCurto()* nessa linha do pseudo-código retorna o primeiro caminho mais curto da rede (N,A), que é então atribuído à p_j.

- **Linha 2:** C é uma fila de prioridades. Comumente, utiliza-se uma *heap* de Fibonacci (FREDMAN; TARJAN, 1987) para armazenar os caminhos candidatos a serem inseridos em L. A *heap* de Fibonacci é utilizada pela necessidade do algoritmo em realizar algumas de suas operações a cada iteração. Essa fila de prioridades possui tempo amortizado constante $\Theta(1)$ em suas operações de *MAKE-HEAP, INSERT, MINIMUM, UNION* e *DECREASE-KEY*, e $O(\log n)$ para as operações de *EXTRACT-MIN* e *DELETE* (CORMEN et al., 2009).

- **Linha 3:** L é a árvore de caminhos determinados. É nela que são armazenados os K caminhos mais curtos encontrados pelo algoritmo de Yen. Um exemplo dessa estrutura pode ser visto em 3.2.

- **Linha 4:** Da linha 4 até a linha 12, a estrutura de repetição é utilizada para buscar os K caminhos mais curtos da rede. Por isso, a execução é pelo valor de K.

- **Linha 5 à 7:** A estrutura condicional dessa linha verifica se ainda existem caminhos candidatos a serem adicionados à L. Se não existir, significa que não existe mais desvios possíveis na rede (N,A) e, por isso, a estrutura de repetição para e retorna L na linha 6.

- **Linha 8:** p_j recebe o caminho no início de C. A função *heapPop()* retorna o menor caminho em C, deletando tal caminho de C.

- **Linha 9:** O caminho p_j é então adicionado à L, se tornando um dos K caminhos mais curtos de (N,A).

- **Linha 10:** Nessa linha, a função *algoritmoDeYen()* é chamada para o caminho p_j e atribuída à heap D_j, que recebe os caminhos de desvio encontrados a partir de p_j. A explicação de como o algoritmo de Yen encontra esses caminhos está no Capítulo 4.

- **Linha 11:** Após obter o conjunto de caminhos de desvio D_j, C recebe a sua união com D_j, para que o processo se repita enquanto $C \neq \varnothing$ ou os K caminhos foram encontrados.

- **Linha 13:** Os K caminhos mais curtos de (N,A) são retornados. Caso a rede não possua K caminhos, os caminhos encontrados são retornados na linha 6.

4 ALGORITMO DE YEN

O algoritmo de Yen (YEN, 1971) é um clássico da literatura do *K-SP*. Sua importância se dá pelo fato de que, diferentemente das soluções propostas anteriormente para o problema, tal abordagem apresentou uma melhor complexidade, com o limite superior computacional de seu algoritmo crescendo linearmente com o valor de K.

O Algoritmo 2 apresenta o pseudo-código do algoritmo de Yen. A explicação do pseudo-código é feita seguindo a numeração de suas linhas.

Algorithm 2 Algoritmo de Yen (YEN, 1971).

Entrada: Rede (N,A), caminho p_j, lista L de caminhos.
Saída: Conjunto D_j de caminhos de desvio.

1: $n_j^m := getVerticeDeDesvio(p_j, L)$
2: $E_j^m := getArcosDeDesvioAssociados(n_j^m)$
3: **for** $i := 1$ **to** $m-1$ **do**
4: $\quad N.removerVertice(n_j^i)$
5: **end for**
6: $A.removerArcosDeDesvio(E_j^m)$
7: **for** $i := m$ **to** $l-1$ **do**
8: $\quad A.removerArco(a_j^i)$
9: $\quad \overline{r}_j^i := (n_j^1, \ldots, n_j^i)$
10: $\quad \overline{s}_j^i := getCaminhoMaisCurto((N,A), s)$
11: $\quad \overline{p}_j^i := getConcatenacao(\overline{r}_j^i, \overline{s}_j^i)$
12: $\quad D_j.heapPush(\overline{p}_j^i)$
13: $\quad N.removerVertice(n_j^i)$
14: **end for**
15: $restaurar(N,A)$
16: **return** D_j

1. **Linha 1:** A função *getVerticeDeDesvio(p_j, L)* é chamada para encontrar o primeiro vértice de desvio do caminho p_j em L e atribuí-lo a n_j^m. Como explicado no Capítulo 3, o primeiro vértice de desvio do caminho p_j é o vértice mais distante da origem, e que coincida com algum vértice de um dos caminhos $j-1$ em L.

2. **Linha 2:** A função *getArcosDeDesvioAssociados(n_j^m)* é chamada, atribuindo a E_m^j os arcos associados ao vértice de desvio n_j^m. Após encontrar o vértice de desvio, os arcos de desvios podem ser facilmente identificados. Para todos os vértices anteriores ao vértice de desvio, inclusive, basta adicionar ao conjunto E_m^j os arcos que conectam esses vértices a qualquer outro vértice em L.

3. **Linha 3:** Da linha 3 a linha 5, os vértices do caminho raiz, que vão do primeiro vértice do caminho p_j até o vértice de desvio são removidos do conjunto N de vértices da rede.

4. **Linha 6:** A função *removerArcosDeDesvio(E_j^m)* é chamada para remover do conjunto A de arcos, os arcos de desvio associados em E_j^m. Dessa forma, evita-se que o algoritmo de Dijkstra chamado na linha 10 encontre caminhos que já foram descobertos em iterações anteriores.

5. **Linha 7:** Da linha 7 a linha 14, sendo m a posição do vértice de desvio n_j^m em p_j, e l a quantidade de arcos de p_j. O algoritmo itera nos vértices do caminho p_j, indo de m a $l-1$, ou seja, a partir do vértice de desvio até o penúltimo vértice de p_j, para calcular os caminhos mais curtos derivados de p_j até o vértice de destino.

6. **Linha 8:** A função *removerArco(a_j^i)* é chamada pelo conjunto A para que seu arco (n_j^i, n_j^{i+1}) seja removido. Assim, para todo o caminho p_j, o algoritmo de Yen passa pelos seus vértices (a partir do vértice de desvio, e sem considerar o vértice de destino) removendo os arcos que conectam o caminho, para que caminhos candidatos a entrarem em L possam ser buscados.

7. **Linha 9:** O caminho raiz $\overline{r}_j^i$ recebe o caminho que vai de n_j^1 a n_j^i.

8. **Linha 10:** A função *getCaminhoMaisCurto(*$(N,A),s$*)* é chamada para calcular o caminho derivado $\overline{s}^i_j$ a ser adicionado no conjunto D_j de caminhos derivados. Assim, a função retorna o caminho mais curto da rede, que não possui os vértices e arcos previamente removidos. Como na linha 1 do Algoritmo 1, o algoritmo de Dijkstra um para todos é utilizado.

9. **Linha 11:** A função *getConcatenacao(*$\overline{r}^i_j$, $\overline{s}^i_j$*)* realiza a concatenação do caminho raiz $\overline{r}^i_j$ com o caminho derivado $\overline{s}^i_j$ calculado na linha anterior, atribuindo o resultado dessa concatenação a $\overline{p}^i_j$.

10. **Linha 12:** O resultado $\overline{p}^i_j$ da linha anterior é inserido na *heap* D_j, passando a fazer parte dos caminhos derivados calculados a partir de p_j.

11. **Linha 13:** O vértice n^i_j é removido do conjunto N de vértices da rede.

12. **Linha 15:** Os vértices e arcos pertencentes a (N,A) são restaurados.

13. **Linha 16:** O conjunto D_j de caminhos derivados calculados para o caminho p_j é retornado ao algoritmo genérico de caminho de desvio.

Para cada execução da linha 10, que utiliza o algoritmo de Dijkstra, no pior caso, têm-se $O(m+n\log n)$ para o cálculo do caminho de desvio candidato. Se for considerado um único caminho p_j a ser analisado pelo algoritimo de Yen, e para essa análise for necessária a verificação dos n vértices da rede, a complexidade passa a ser $O(n(m+n\log n))$. Dessa forma, para os K caminhos mais curtos que devem ser encontrados, sabendo que para cada caminho em L, o algoritmo de Yen é chamado uma vez, a complexidade no pior caso passa a ser $O(Kn(m+n\log n))$.

5 RESULTADOS E TESTES

A instância utilizada para teste é a descrita na Figura 3.1. O objetivo do trabalho é ser um facilitador para o entendimento do problema dos K caminhos mais curtos, o conceito de algoritmo de desvio e do algoritmo de Yen (YEN, 1971). Dessa forma, os resultados aqui apresentados tem a intenção de exemplificar as iterações do algoritmo. A implementação[1] foi feita utilizando a linguagem de programação Python (versão 3.6.9). Para a *heap* de Fibonacci, foi utilizada uma biblioteca[2].

As Figuras 5.1 a 5.4 exemplificam as iterações do algoritmo de Yen. Cada uma delas inicia com a exibição do caminho p_j atual. Como explicado no Capítulo 3, os caminhos mais curtos são adicionados à estrutura L a cada iteração do algoritmo genérico de caminho de desvio para os K caminhos mais curtos a serem encontrados, ou até não existir mais caminhos possíveis na rede (N,A).

O algoritmo genérico de caminho de desvio inicia com o caminho mais curto da rede já calculado. Assim, na Figura 5.1, é possível observar que o algoritmo de Yen é iniciado com o p_1 já determinado. Além disso, nessa primeira iteração do algoritmo, dois caminhos mais curtos, $\langle v_1,(v_1,v_2),v_2,(v_2,v_4),v_4,(v_4,v_{10}),v_{10}\rangle$ e $\langle v_1,(v_1,v_3),v_3,(v_3,v_6),v_6,(v_6,v_8),v_8,(v_8,v_7),v_7,(v_7,v_{10}),v_{10}\rangle$ (que aparecem no resultado final na Figura 5.5), já são encontrados e adicionados ao conjunto de caminhos derivados obtidos pelo algoritmo de Yen, D_j. Após terminar a execução da primeira iteração do algoritmo de Yen, uma *heap* de Fibonacci é retornada para o algoritmo de desvio. Na sequência, efetuamos a união de D_j à heap C (operação *UNION*) e atualizamos a estrutura C.

No início da segunda iteração do algoritmo genérico de caminho de desvio, antes da iteração apresentada na Figura 5.2, C contém os caminhos $\langle v_1,(v_1,v_2),v_2,(v_2,v_4),v_4,(v_4,v_{10}),v_{10}\rangle$ e $\langle v_1,(v_1,v_3),v_3,(v_3,v_6),v_6,(v_6,v_8),v_8,(v_8,v_7),v_7,(v_7,$

[1] <https://github.com/quionee/k-shortest-path>
[2] <https://pypi.org/project/fibheap/>

$v_{10}\rangle, v_{10}\rangle$. A iteração 2, Figura 5.2, começa com o p_j extraído do início de C. Para essa iteração, mais nenhum caminho de desvio é encontrado pelo algoritmo de Dijkstra. Isso se deve ao fato de que o único vértice do caminho p_2, $\langle v_1, (v_1, v_2), v_2, (v_2, v_4), v_4, (v_4, v_{10}), v_{10}\rangle$, que coincide com outros vértices da rede, ser o vértice v_1. Porém, como pode ser visto na exibição do campo *Arcos da rede* em 5.2a, o arco (v_1, v_3) foi retirado da rede por ser um arco de desvio associado ao vértice v_1, que é o primeiro vértice de desvio para essa iteração do algoritmo de Yen. Assim, nessa iteração, mais nenhum caminho de desvio é adicionado a C.

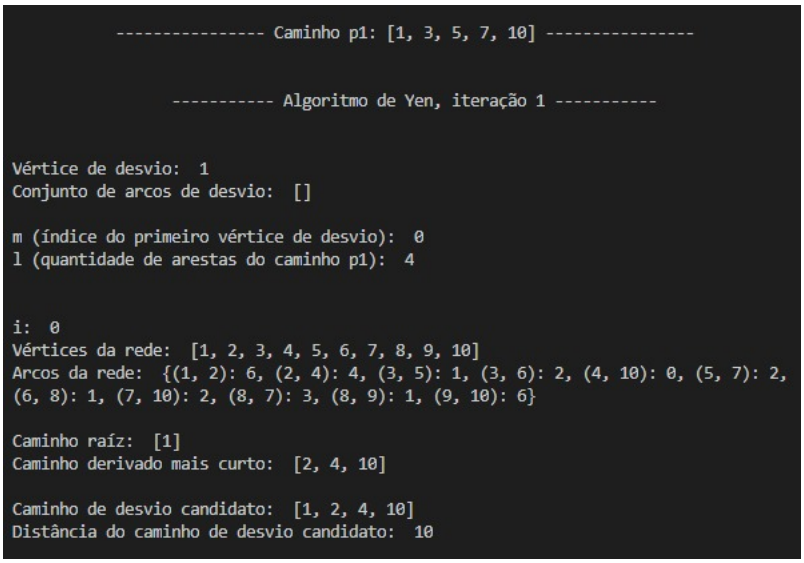
Um ponto vísivel nas iterações do algoritmo de Yen é relacionado às remoções de arcos e vértices na rede (N,A). Isso ocorre para que o algoritmo de Dijkstra seja capaz de encontrar novos caminhos mais curtos, sem repetir caminhos já selecionados, ou mesmo ficar "preso" nestes caminhos. Essas remoções podem ser observadas nas figuras pelo campo *Vértices da rede* e *Arcos da rede*.

Na terceira iteração, o vértice de desvio é o vértice v_3, como exemplificado na Figura 3.3c. A partir do caminho p_3, o quarto caminho de desvio é encontrado, como pode ser visto na Figura 5.3c. O caminho de desvio candidato encontrado, exibido no campo *Caminho de desvio candidato*, será adicionado a C e posteriormente a L. Na última iteração do Algoritmo de Yen, mais nenhum caminho é encontrado na rede apresentada na Figura 3.1.

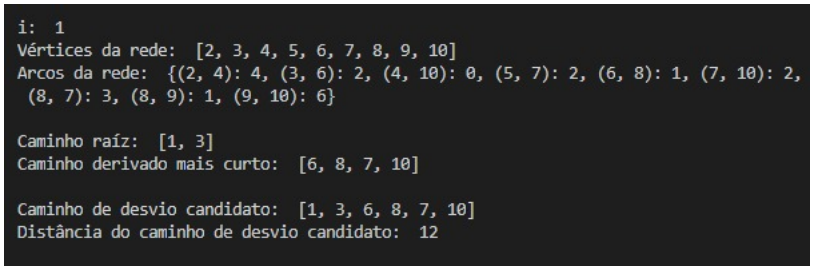
Por fim, os K caminhos mais curtos da rede são apresentados na Figura 5.5. A estrutura é armazenada como uma pseudo-árvore, assim, para o vértice v_1, por exemplo, sabe-se sua quantidade de ramificações e em quais vértices essas ramificações se conectam. Assim, para o caminho p_3, por exemplo, sabemos, percorrendo a estrutura, que ele é composto dos vértices v_1, v_3, v_6, v_8, v_7 e v_{10}. A Figura 5.5 pode ser analisada em conjunto com a Figura 3.2 para um melhor entendimento.

Figura 5.1 – Iteração 1 do Algoritmo de Yen.

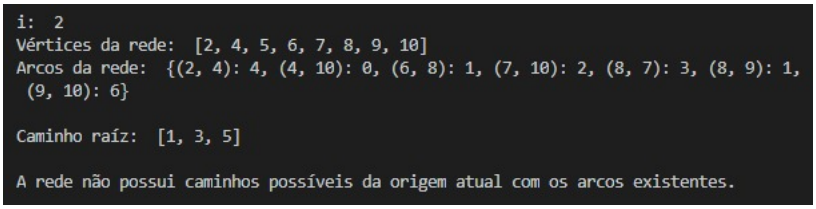
(a) Parte 1.



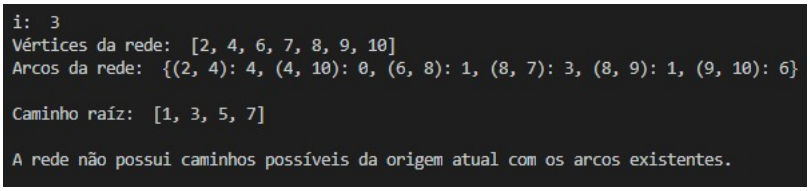
(b) Parte 2.



(c) Parte 3.



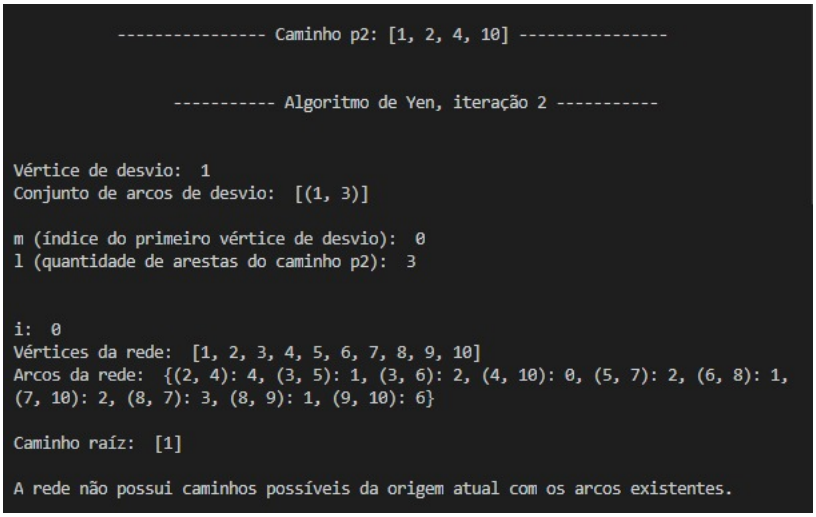
(d) Parte 4.



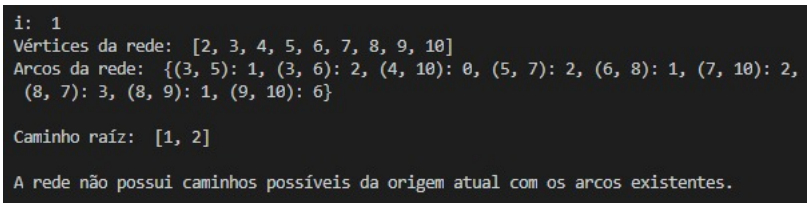
Fonte: elaborado pela autora.

Figura 5.2 – Iteração 2 do Algoritmo de Yen.

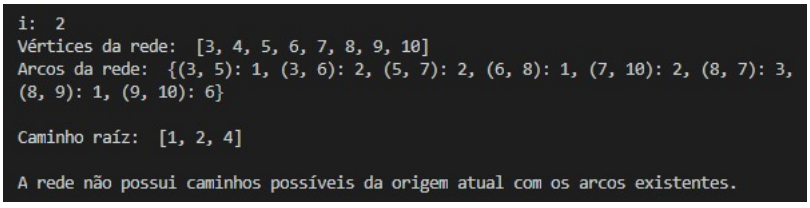
(a) Parte 1.



(b) Parte 2.



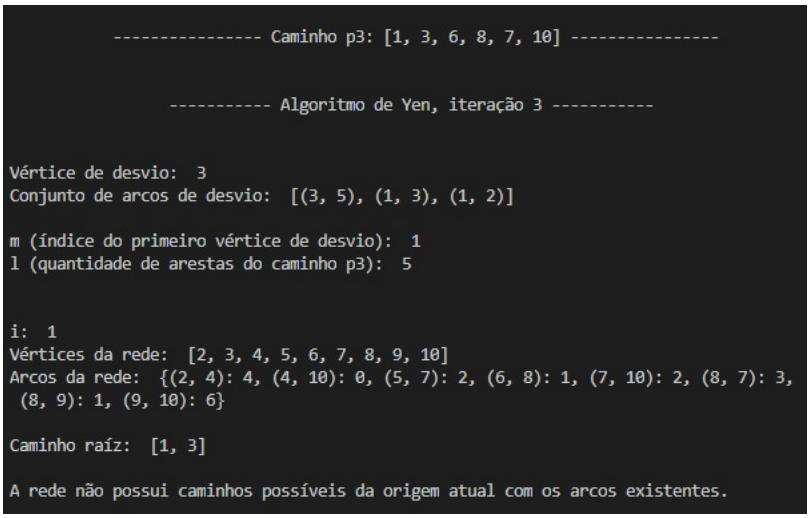
(c) Parte 3.



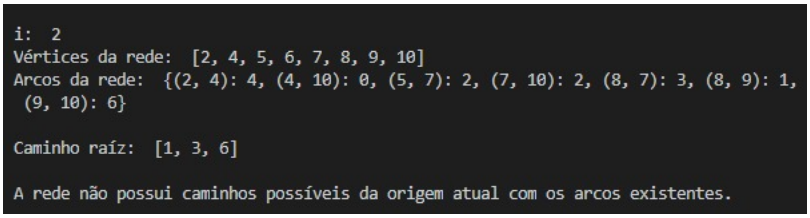
Fonte: elaborado pela autora.

Figura 5.3 – Iteração 3 do Algoritmo de Yen.

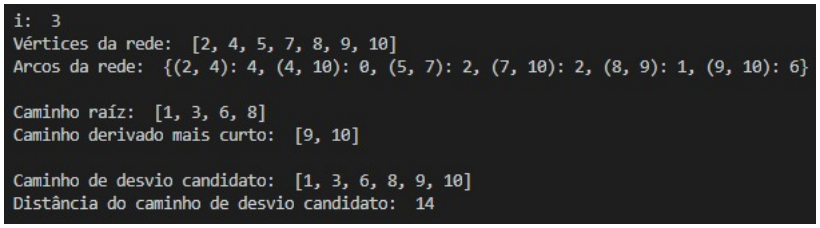
(a) Parte 1.



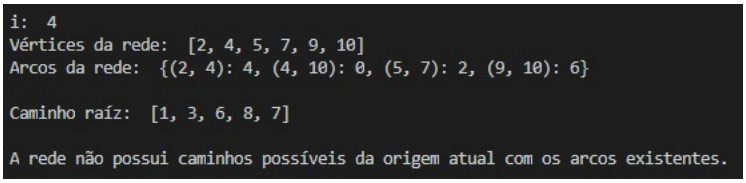
(b) Parte 2.



(c) Parte 3.



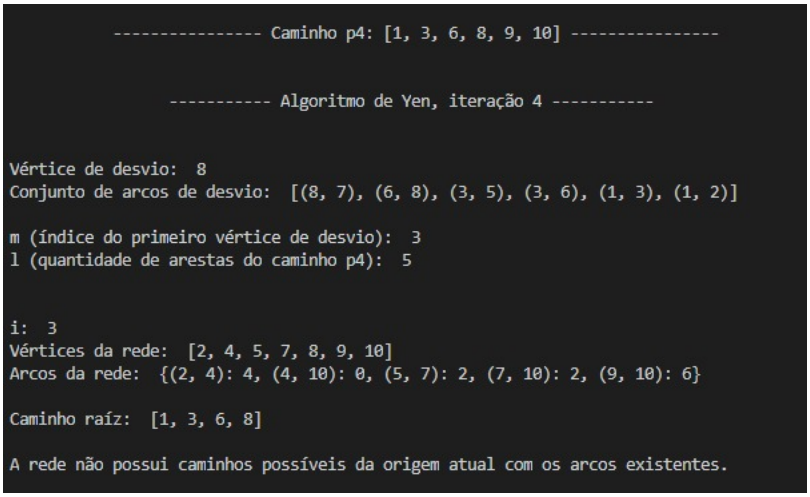
(d) Parte 4.



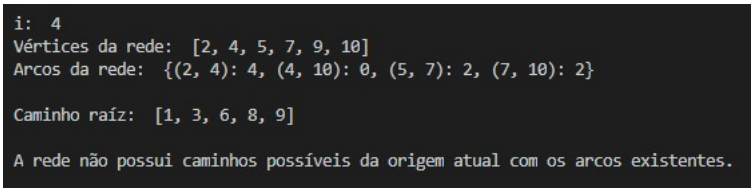
Fonte: elaborado pela autora.

Figura 5.4 – Iteração 4 do Algoritmo de Yen.

(a) Parte 1.



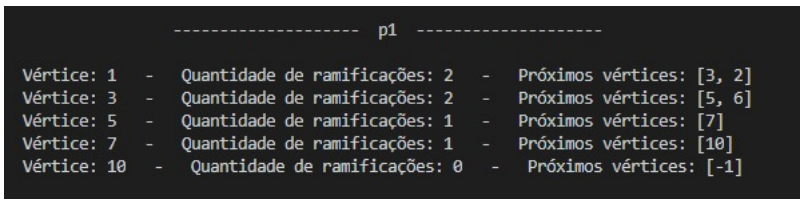
(b) Parte 2.



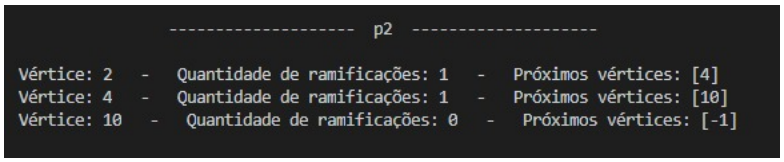
Fonte: elaborado pela autora.

Figura 5.5 – Pseudo-árvore do resultado com os $K = 4$ caminhos mais curtos da rede (N,A).

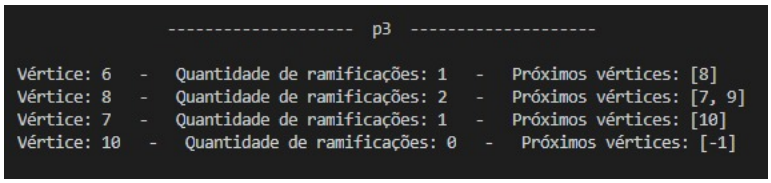
(a) Caminho p1.



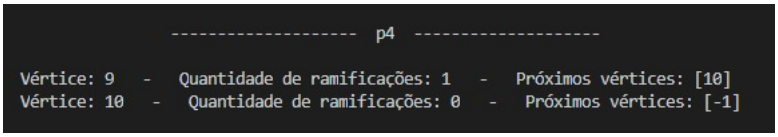
(b) Caminho p2.



(c) Caminho p3.



(d) Caminho p4.



Fonte: elaborado pela autora.

6 CONCLUSÃO

O objetivo deste documento foi realizar um estudo sobre o problema dos K caminhos mais curtos acíclicos, apresentando a explicação de um algoritmo clássico da literatura para o problema, o algoritmo de Yen. A explicação do conceito de algoritmos de desvio e do algoritmo de Yen foram desenvolvidas com o objetivo de auxiliar o entendimento do tema para a comunidade acadêmica interessada. Além disso, o problema possui bastante relevância prática, podendo ser aplicado em diversas situações.

Para trabalhos futuros, a metodologia utilizada neste trabalho - implementação e explicação detalhada do algoritmo - pode ser aplicada a outros algoritmos desenvolvidos para o problema dos K caminhos mais curtos. Um possível algoritmo é apresentado em (MARTINS; PASCOAL, 2003) (algoritmo de MP), em que os autores apresentam uma nova implementação do algoritmo de Yen. Outro algoritmo pode ser visto em (CHEN et al., 2020). Neste, os autores apresentam o algoritmo de MP e o algoritmo de Yen, e utilizam, além do conceito de algoritmo de desvio, uma técnica de reotimização com A*, incorporada ao algoritmo para o cálculo do caminho de desvio.

A implementação disponibilizada pelo trabalho pode ser expandida para a implementação de ambos os algoritmos mencionados acima. Além disso, o uso do pacote NetworkX[1] para geração de grafos e estruturas acrescentaria uma visualização ainda maior para os resultados.

[1] <https://networkx.org/>

REFERÊNCIAS

CHEN, B. Y. et al. Efficient algorithm for finding k shortest paths based on re-optimization technique. **Transportation Research Part E: Logistics and Transportation Review**, v. 133, 2020. ISSN 13665545.

CORMEN, T. H. et al. **Introduction to Algorithms**. 3. ed. [S.l.]: MIT Press, 2009. ISBN 9780262033848.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, n. 1, 1959. ISSN 0029599X.

EPPSTEIN, D. Finding the k shortest paths. **SIAM Journal on Computing**, v. 28, n. 2, 1998. ISSN 00975397.

FREDMAN, M. L.; TARJAN, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. **Journal of the ACM (JACM)**, v. 34, n. 3, 1987. ISSN 1557735X.

HOFFMAN, W.; PAVLEY, R. A Method for the Solution of the Nth Best Path Problem. **Journal of the ACM (JACM)**, v. 6, n. 4, 1959. ISSN 1557735X.

MARTINS, E. Q.; PASCOAL, M. M. A new implementation of Yen's ranking loopless paths algorithm. **4OR**, v. 1, n. 2, 2003. ISSN 16142411.

VANHOVE, S.; FACK, V. An effective heuristic for computing many shortest path alternatives in road networks. **International Journal of Geographical Information Science**, v. 26, n. 6, 2012. ISSN 13623087.

YEN, J. Y. Finding the K Shortest Loopless Paths in a Network . **Management Science**, v. 17, n. 11, 1971. ISSN 0025-1909.