k-MENORES CAMINHOS

FÁBIO PISARUK

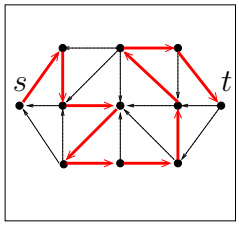
Orientador: José Coelho de Pina

RESUMO. Neste projeto, fazemos um esboço de uma dissertação de mestrado que pretende estudar algoritmos desenvolvidos para geração de k-menores caminhos em grafos com custos não-negativos, bem como algumas implementações destes.

SUMÁRIO

1. INTRODUÇÃO

[l]30mm



Uma certa empresa de telecomunicações, cujo nome real não será citado por razões de confidencialidade, mas que para nossa comodidade será chamada de TeleMax, fornece linhas de transmissão aos seus clientes de modo que estes possam, por exemplo, ligar-se às suas filiais por linhas privadas. Para tal, conta com uma infra-estrutura (rede) bastante complexa compreendendo cabos e diversos equipamentos de junção. Esta rede é *full-duplex*, ou seja, possui passagem de dados em ambos os sentidos. Para entender o processo de fornecimento de linhas de transmissão, passaremos a um exemplo. A empresa SoftSOF possui duas sedes, uma em Santos e outra em

Data: 23 de julho de 2012.
Key words and phrases. menores caminhos, caminhos mínimos, otimização combinatória.

Fernandópolis e, deseja interligar suas filiais com uma qualidade mínima de 200 Kbits/s, em pico de uso. A SoftSOF possui 10 computadores em cada uma de suas filiais, logo precisaríamos de um link de 200 Kbits/s * 10 = 2000 Kbits/s. É solicitado à TeleMax um *link* de 2000 Kbits/s ligando as suas sedes. A TeleMax não possui uma ligação direta entre as duas cidades, entretanto possui uma ligação que passa por São José do Rio Preto, ou seja, um caminho Santos - São José Do Rio Preto - Fernandópolis. Infelizmente, este link só dispõe de 1024 Kbits/s. No entanto, observando-se sua infra-estrutura, descobre-se que existe um outro caminho: Santos - São Paulo - Fernandópolis, também com capacidade de 1024 Kbits/s. Pronto. A TeleMax pode fornecer o link requerido pela SoftSOF, bastando para isso utilizar os dois caminhos acima descritos, totalizando os 2048 Kbits/s, um pouco acima do requerido.

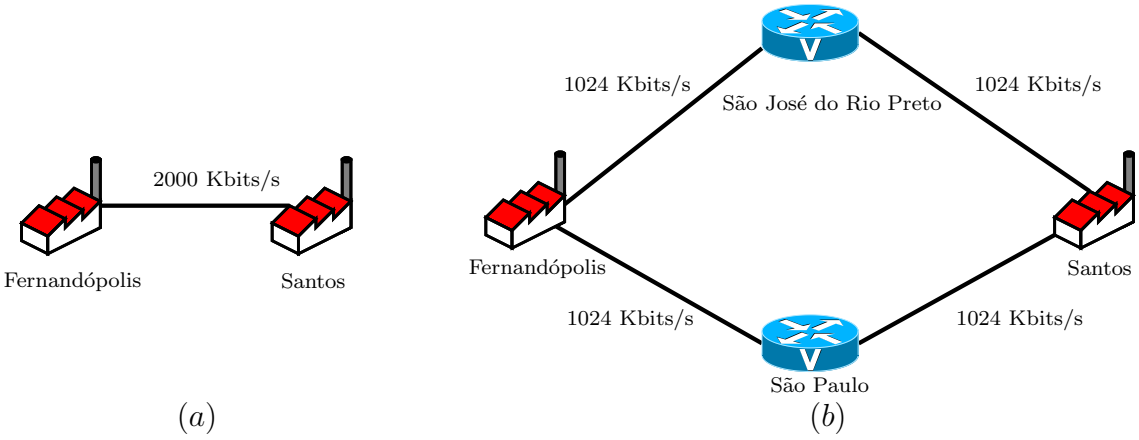


FIGURA 1. Em (a) vemos um esquema solicitação da SoftSOF, um link de 2000 Kbits/s entre as filiais. Em (b) está descrita a solução encontrada pela TeleMax, com base na disponibilidade de sua rede.

Vamos a algumas considerações relevantes. O custo de um caminho é função da quantidade de equipamentos usada e não da distância total dos cabos que o compõe. Isto se deve ao custo elevado dos equipamentos se comparado ao dos cabos. Assim, passa a ser melhor utilizar uma ligação que percorra uma distância maior mas que passa por um número menor de equipamentos, do que uma com menor distância mas que se utiliza de mais equipamentos.

A justificativa para a geração de diversos caminhos no lugar de apenas um está relacionada à capacidade de transmissão disponível por cabo. A motivação para a geração dos menores caminhos, ou seja, com utilização mínima de equipamentos, requer uma explicação mais detalhada. Até agora fomos simplistas ao tratarmos das relações entre cabos e equipamentos como se um equipamento se ligasse a apenas um cabo. Na verdade, cada equipamento se liga a um grande número de cabos. Assim, podemos ter diversos caminhos entre dois equipamentos, um para cada cabo. A

fim de utilizarmos bem os recursos da rede é interessante que o menor número de equipamentos esteja alocado para cada cliente pois, desta maneira, um número maior de ligações poderá ser oferecido pela TeleMax. Embora a utilização do menor número possível de equipamentos para cada cliente não seja suficiente para garantir que a rede esteja sendo utilizada de maneira eficiente, não nos importaremos com isto neste trabalho. Feitas as devidas considerações, vamos agora justificar a automação do processo.

Imagine levar a cabo o processo de fornecimento de linhas manualmente. Podemos salientar alguns problemas da abordagem manual. Devido às dimensões da rede, o operador responsável levará muito tempo para obter uma lista de caminhos entre os pontos. Durante o tempo em que o operador gastar analisando a rede, esta poderá ter sofrido alterações, as quais não serão levadas em conta por ele. Além disso, sabemos como as pessoas são suscetíveis a falhas, ainda mais quando expostas a atividades maçantes e repetitivas. Por conta destes fatores, a TeleMax sentiu a necessidade de uma ferramenta computacional que gerasse de maneira rápida e confiável uma série de caminhos entre dois pontos da sua rede.

Na construção da ferramenta, consideramos a rede como um grafo simétrico, por ser full-duplex, onde as arestas são representadas pelos cabos e os vértices pelos equipamentos. A ferramenta tinha como núcleo o algoritmo desenvolvido por Naoki Katoh, Toshihide Ibaraki e H. Mine [8], de geração de menores caminhos. Os caminhos de mesmo custo, ou seja, que se utilizam de igual quantidade de equipamentos, são posteriormente reordenados crescentemente pela distância total percorrida por seus cabos. Esta dissertação trata de algoritmos que produzem caminhos de menor custo em grafos. Embora algoritmos para tal sejam de interesse teórico, é curioso observar que foi uma aplicação prática, demandada por uma necessidade surgida no âmbito empresarial, que nos levou ao estudo destes.

2. Grafos, passeios e caminhos

Estão descritos nesta seção os ingredientes básicos que envolvem os problemas que serão tratados. A notação básica que utilizamos é a de Paulo Feofillof [4]

Um **grafo** é um objeto da forma (V, A), onde V é um conjunto finito e A é um conjunto de pares ordenados de elementos de V.

Os elementos de V são chamados **vértices** e os elementos de A são chamados **arcos**. Para cada arco (u, v), os vértices u e v representam a ponta inicial e a ponta final de (u, v), respectivamente. Um arco (u, v) também poderá ser representado por uv.

Um grafo é **simétrico** se para cada arco uv existir também o arco vu. Diremos às vezes que o arco vu é **reverso** do arco uv e que o par $\{(u, v), (v, u)\}$ é uma **aresta**.

Um grafo pode ser naturalmente representado através de um diagrama, como o da figura 2, onde os vértices são pequenas bolas e os arcos são as flechas ligando estas bolas.

Um **passeio** num grafo (V, A) é qualquer seqüência da forma

$$\langle v_0, a_1, v_1, \ldots, a_t, v_t \rangle \tag{1}$$

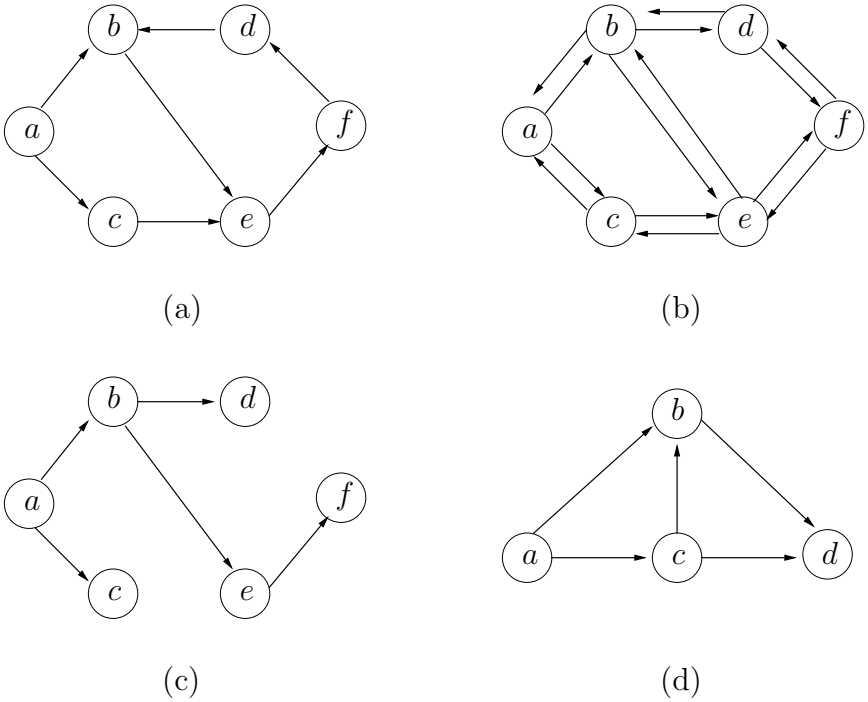


FIGURA 2. (a), (b), (c) e (d) são exemplos de grafos. (b) é um grafo simétrico.

onde $v_0, \ldots, v_t$ são vértices, $a_1, \ldots, a_t$ são arcos e, para cada i, a_i é o arco $v_{i-1}v_i$. O vértice v_0 é o **início** ou **ponta inicial** do passeio e o v_t é seu **término** ou **ponta final**. Na figura 2(a) a seqüência $\langle a, ab, b, be, e, ef, f, fd, d, db, b, be, e, ef, f\rangle$ é um passeio com início em a e término em f.

Se $P := \langle v_0, a_1, v_1, \ldots, a_t, v_t\rangle$, então qualquer subseqüência da forma

$$\langle v_i, a_{i+1}, v_{i+1}, \ldots, a_j, v_j\rangle \tag{2}$$

com $0 \leq i \leq j \leq t$ será um **sub-passeio** de P. Além disso, se $i = 0$, então o sub-passeio será dito um **prefixo** de P. Na figura 2(a) a seqüência $\langle a, ab, b, be, e, ef, f, fd, d\rangle$ é um sub-passeio e prefixo de do passeio $\langle a, ab, b, be, e, ef, f, fd, d, db, b, be, e, ef, f\rangle$.

Um **caminho** é um passeio sem vértices repetidos. Na figura 2(a) a seqüência $\langle a, ab, b, be, e, ef, f\rangle$ é um caminho com início em a e término em f.

Uma **função custo** em (V, A) é uma função de A em $\mathbb{Z}_{\geq}$. Se c for uma função custo em (V, A) e uv estiver em A, então $c(u, v)$ será o valor de c em uv. Se P for um passeio em um grafo (V, A) e c uma função custo, denotaremos por $c(P)$ o **custo do caminho** P, ou seja, $c(P)$ é o somatório dos custos de todos os arcos em P. Um passeio P tem **custo mínimo** se $c(P) \leq c(P')$ para todo passeio P' que tenha o mesmo início e término que P. Um passeio de custo mínimo é comumente chamado de **caminho mínimo**.

Um problema fundamental em otimização combinatória que tem um papel de destaque neste projeto é o **problema do caminho mínimo**, denotado por CM:

CM **Problema** CM(V, A, c, s, t): Dado um grafo (V, A), uma função custo c

e dois vértice s e t, encontrar um caminho de custo mínimo de s a t.

Na literatura essa versão é conhecida como *single-pair shortest path problem*. O celebrado algoritmo de Edsger Wybe Dijkstra [2] resolve o problema do caminho mínimo.

Denotaremos, quando não houver ambigüidade, por n e m os números $|V|$ e $|A|$, respectivamente. Além disso, representaremos por $T(n, m)$ o consumo de tempo de uma subrotina genérica para resolver o CM em um grafo com n vértices e m arestas. O algoritmo mais eficiente conhecido para o CM foi projetado por Michael L. Fredman e Robert Endre Tarjan [5] e consome tempo $\mathrm{O}(m + n \log n)$. Existe ainda um algoritmo que consome tempo linear *sob um outro modelo de computação* que foi desenvolvido por Mikkel Thorup [15].

3. Árvores dos prefixos

Descrevemos aqui uma "arborescência rotulada" que de certa forma codifica os prefixos dos caminhos em uma dada coleção. Esta representação será particularmente útil quando, mais adiante, discutirmos o método de Yen. No que segue $\mathcal{Q}$ é uma coleção de caminhos de um grafo e $V(\mathcal{Q})$ e $A(\mathcal{Q})$ são o conjunto dos vértices e o conjunto dos arcos presentes nos caminhos, respectivamente.

Um grafo acíclico (N, E) com $|N| = |E|+1$ é uma **arborescência** se todo vértice, exceto um vértice especial chamado de **raiz**, for ponta final de exatamente um arco. Será conveniente tratarmos os vértices de uma arborescência por **nós**. Uma arborescência está ilustrada na figura 2(c). A raiz dessa arborescência é o nó a. Uma **folha** de uma arborescência é um nó que não é ponta inicial de nenhum arco.

Suponha que (N, E) seja uma arborescência e f uma **função rótulo** que associa a cada nó em N um vértice em $V(\mathcal{Q})$ e a cada arco em E um arco em $A(\mathcal{Q})$. Se

$$R = \langle u_0, e_1, u_1, \ldots, e_t, u_t \rangle$$

for um caminho em (N, E), então

$$f(R) := \langle f(u_0), f(e_1), f(u_1), \ldots, f(e_t), f(u_t) \rangle$$

será uma seqüência de vértices e arcos dos caminhos em $\mathcal{Q}$. Diremos que (N, E, f) é **árvore dos prefixos** de $\mathcal{Q}$ se

(p1) para cada caminho R em (N, E) com início na raiz, $f(R)$ for prefixo de algum caminho em $\mathcal{Q}$; e
(p2) para cada prefixo Q de algum caminho em $\mathcal{Q}$ existir um caminho R em (N, E) com início na raiz tal que $f(R) = Q$; e
(p3) o caminho R do item anterior for único.

Não é verdade que para cada coleção $\mathcal{Q}$ de caminhos em um grafo existe uma árvore dos prefixos de $\mathcal{Q}$. No entanto, se todos os caminhos em $\mathcal{Q}$ tiverem a mesma ponta inicial, então existe uma árvore dos prefixo de $\mathcal{Q}$ e esta é única. Na figura 3(b) vemos a ilustração da árvore dos prefixos de quatro caminhos de s a t no grafo da figura 3(a). Na árvore da ilustração w, x, y e z são nós e $f(w) = s, f(x) = a, f(y) = d$ e $f(z) = d$.

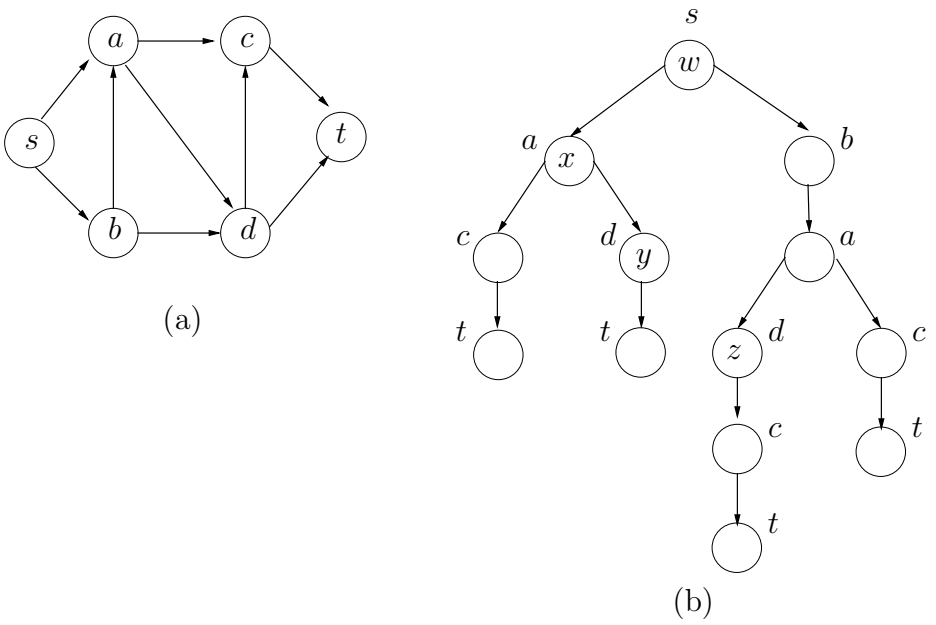


FIGURA 3. (b) mostra a árvore dos prefixos dos caminhos $\langle s, a, c, t\rangle$, $\langle s, a, d, t\rangle$, $\langle s, b, a, c, t\rangle$ e $\langle s, b, a, d, c, t\rangle$ no grafo em (a). Na árvore, um símbolo ao lado de um nó é o rótulo desse nó. Os rótulos dos arcos não estão representados na figura. O símbolo dentro de um nó é o seu nome.

4. Problema dos k-caminhos

O problema central deste projeto se assemelha muito ao do k-ésimo menor elemento, que é estudado em disciplinas básicas de análise de algoritmos:

k-ÉSIMO

> **Problema** k-ÉSIMO$(\mathcal{S}, k)$: Dado um conjunto $\mathcal{S}$ de números inteiros e um número inteiro positivo k, encontrar o k-ésimo menor elemento de $\mathcal{S}$.

Os algoritmos conhecidos para o problema k-ÉSIMO são facilmente adaptáveis para, além do k-ésimo menor, fornecerem, em tempo linear, os k menores elementos de $\mathcal{S}$ *em ordem crescente*.

A diferença entre o problema k-ÉSIMO e o problema que consideraremos é que o conjunto $\mathcal{S}$ dado é "muito grande" e, portanto, nos é dado de uma maneira compacta, o que torna o problema sensivelmente mais difícil do ponto de vista computacional. Adiante tornamos o problema mais preciso.

Suponha que (V, A) seja um grafo, c uma função custo e s e t dois de seus vértices. Considere o conjunto $\mathcal{P}_{st}$ de todos st-caminhos, ou seja, caminhos de s a t. Uma lista $\langle P_1, \ldots, P_k\rangle$ st-caminhos distintos é de **custo mínimo** se

$$c(P_1) \leq c(P_2) \leq \cdots \leq c(P_k) \leq \min\{c(P) : P \in \mathcal{P} - \{P_1, \ldots, P_k\}\}.$$

De uma maneira mais breve, diremos que $\langle P_1, \ldots, P_k\rangle$ são **k-menores caminhos** (de s a t).

Em termos da teoria dos grafos o problema que foi discutido na introdução é o **problema dos k-menores caminhos**, denotado por k-CM:

> **Problema** k-CM(V, A, c, s, t, k): Dado um grafo (V, A), uma função custo c, dois vértice s e t e um inteiro positivo k, encontrar k-caminhos de custos mínimos de s a t. k-CM

É evidente que o CM nada mais é que o k-CM com $k = 1$.

O k-CM é, em essência, o problema k-ÉSIMO com $\mathcal{P}_{st}$ no papel do conjunto $\mathcal{S}$. A grande diferença computacional é que o conjunto $\mathcal{P}_{st}$ não é fornecido explicitamente, mas sim de uma maneira compacta: um grafo, uma função custo e um par de vértices. Desta forma, o número de elementos em $\mathcal{P}_{st}$ é potencialmente exponencial no tamanho da entrada, tornando impraticável resolvermos o k-CM utilizando meramente algoritmos para o k-ÉSIMO como subrotina.

Na próxima seção é descrito o método genérico para resolver o k-CM. Este método é um passo intermediário para chegarmos no método desenvolvido por Jin Y. Yen [16] para o k-CM. Antes disto, apresentamos aqui um problema intimamente relacionado ao k-CM e que também é considerado por este projeto:

> **Problema** K-CM(V, A, c, s, t, K): Dado um grafo (V, A), uma função custo c, dois vértice s e t e um inteiro positivo K, encontrar os caminhos de s a t de custos não superiores a K. K-CM

5. Método genérico

A descrição que fazemos é, de certa forma, *top-down*. Começaremos com um método genérico que será refinado a cada passo incluindo, convenientemente, algumas subrotinas auxiliares. O nosso interesse aqui é numa descrição mais conceitual em que a correção e o consumo de tempo polinomial do método sejam um tanto quanto evidentes. Não temos a intenção de descrever um algoritmo com o menor consumo de tempo.

O método abaixo recebe um grafo (V, A), uma função custo, dois vértices s e t e um inteiro positivo k e devolve uma lista $\langle P_1, \ldots, P_k \rangle$ de k-menores caminhos de s a t.

Método GENÉRICO (V, A, c, s, t, k)

```
0   P ← conjunto dos caminhos de s a t
1   para i = 1, ..., k faça
2       P_i ← caminho de custo mínimo em P
3       P ← P − P_i
4   devolva ⟨P_1, ..., P_k⟩
```

No início de cada iteração da linha 1 o conjunto $\mathcal{P}$ contém os candidatos a i-ésimo caminho mínimo de s a t. O método de Yen é uma elaboração do método GENÉRICO. Em vez do conjunto $\mathcal{P}$, Yen mantém, pelo menos conceitualmente, uma partição Π de $\mathcal{P}$. Em cada iteração, é escolhido o caminho mais barato dentre um

conjunto $\mathcal{L}$ formado por *um* caminho mínimo P_π representante de cada parte π de Π e depois a partição é atualizada.

Método YEN-GENÉRICO (V, A, c, s, t, k)

0 $\Pi \leftarrow \{\{$*conjunto dos caminhos de* s *a* $t\}\}$

1 $\mathcal{Q} \leftarrow \emptyset$

2 ***para*** $i = 1, \ldots, k$ ***faça***

3 $\quad \mathcal{L} \leftarrow \langle P_\pi : P_\pi$ *é caminho mínimo da parte* π *de* $\Pi\rangle$

4 $\quad P_i \leftarrow$ *caminho de custo mínimo em* $\mathcal{L}$

5 $\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{P_i\}$

6 $\quad \Pi \leftarrow$ ATUALIZE-GENÉRICO $(V, A, \mathcal{Q})$

7 ***devolva*** $\langle P_1, \ldots, P_k\rangle$

Como veremos, a eficiência do método de Yen dependerá fortemente da estrutura restrita dos caminhos nas partes de Π: cada parte é formada por caminhos que têm um certo prefixo comum.

Seja $\mathcal{P}_{st}$ a coleção dos caminhos de s a t em (V, A). Suponha que $\mathcal{Q}$ seja a lista de caminhos distintos de s a t na linha 5 do método YEN-GENÉRICO. Passamos a descrever a partição Π dos caminhos em $\mathcal{P} := \mathcal{P}_{st} \setminus \mathcal{Q}$. Para isto é conveniente utilizarmos a *árvore dos prefixos* de $\mathcal{Q}$, como foi feito por John Hershberger, Matthew Maxel e Subhash Suri [7].

No que segue suponha que (N, E, f) seja a árvore dos prefixos de $\mathcal{Q}$ e u seja um
R_u nó em N. Representaremos por R_u o caminho da raiz a u na árvore. Assim, $f(R_u)$ é
$f(R_u)$ o prefixo de um caminho em $\mathcal{Q}$. Por exemplo, na árvore dos prefixos da figura 3(b) temos que $R_y = \langle w, wx, x, xy, y\rangle$ e $f(R_y) = \langle s, sa, a, ad, d\rangle$.

A_u Seja

$$A_u := \{(f(u), f(w)) : uw \in E\}.$$

π_u e seja π_u o conjunto dos caminhos em $\mathcal{P}$ com prefixo $f(R_u)$ e que não possuem arcos em A_u. Para o exemplo na figura 3 temos que

$$A_w = \{sa, sb\}, A_x = \{ac, ad\}, A_y = \{dt\} \text{ e } A_z = \{dc\}$$
$$\pi_w = \emptyset, \pi_x = \emptyset, \pi_y = \{\langle s, a, d, c, t\rangle\}, \text{ e } \pi_z = \{\langle s, b, a, d, t\rangle\}.$$

Π A partição Π é formada por uma parte π_u para cada vértice u em N, ou seja,

$$\Pi := \{\pi_u : u \in N\}.$$

No início de cada iteração da linha 2 o número de partes é certamente não superior a $n \times i$. O algoritmo ATUALIZE-GENÉRICO resume toda a discussão acima.

Algoritmo ATUALIZE-GENÉRICO $(V, A, \mathcal{Q})$

0 $\Pi \leftarrow \emptyset \qquad \mathcal{P} \leftarrow \mathcal{P}_{st} \setminus \mathcal{Q}$

1 $(N, E, f) \leftarrow$ *árvore dos prefixos de* $\mathcal{Q}$

2 ***para cada*** $u \in N$ ***faça***

3 $\quad \pi_u \leftarrow \{$*caminhos em* $\mathcal{P}$ *com prefixo* $f(R_u)$
e que não possuem arcos em $A_u\}$

4 $\quad \Pi \leftarrow \Pi \cup \{\pi_u\}$

5 ***devolva*** Π

Podemos verificar que cada caminho em $\mathcal{Q}$ não pertence a nenhuma parte de Π. Também podemos verificar que cada caminho P em $\mathcal{P}$ está em uma única parte de Π. De fato, seja P' o maior prefixo de P que é prefixo de algum caminho em $\mathcal{Q}$. Pela definição de árvore de prefixos, existe um único caminho R' em (N, E) com início na raiz e tal que $P' = f(R')$. Para o vértice u término de R' temos que P está em π_u e é a única parte que possui P.

Desta forma, no início de cada iteração das linhas 2–6 do método Yen-Genérico, Π é uma partição de $\mathcal{P}$, portanto a correção do método é evidente.

As árvores dos prefixos de duas execuções consecutivas do algoritmo Atualize-Genérico são muito semelhantes: apenas um novo caminho é acrescentado à árvore anterior. Isto, em particular, significa que as partições de duas iterações consecutivas das linhas 2–6 do método Yen-Genérico são muitos semelhantes. Esta observação pode ser utilizada para o algoritmo Atualize-Genérico obter mais eficientemente uma partição a partir da partição anterior.

6. Método de Yen

O método que Jin Y. Yen [16] desenvolveu para resolver o k-CM parece ter um papel central entre os algoritmos que foram posteriormente projetados para o k-CM ou mesmo para versões mais restritas do problema [3, 8, 7]. Várias melhorias práticas do método de Yen têm sido implementadas e testadas [1, 6, 11, 12, 14]

Antes de prosseguirmos, mencionamos que o método de Yen foi generalizado por Eugene L. Lawler [9] para problemas de otimização combinatória, contanto que seja fornecida uma subrotina para determinar uma solução ótima sujeita a condição de que certas variáveis têm seus valores fixados. Por exemplo, no caso do método de Yen para o k-CM essa subrotina resolve o seguinte **problema do sub-caminho mínimo**, denotado por SCM:

> **Problema** SCM(V, A, c, s, t, P, F): Dado um grafo (V, A), uma função custo c, dois vértice s e t, um caminho P e uma parte F de A, encontrar um caminho de custo mínimo de s a t que tem P como prefixo e não contém arcos em F. SCM

É evidente que se P não tem início em s então o problema é inviável. Do ponto de vista de método Lawler, o prefixo P e o conjunto F são as 'variáveis' com valores fixados.

Resolver o CM(V, A, c, s, t) é o mesmo que resolver SCM$(V, A, c, s, t, \langle s\rangle, \emptyset)$. Por outro lado, o SCM pode ser solucionado aplicando-se um algoritmo para o CM em um sub-grafo apropriado de (V, A). Desta forma, o CM e o SCM são computacionalmente equivalentes e podem ser resolvidos em tempo $T(n, m)$.

Conceitualmente, o método de Yen é uma elaboração do método Yen-Genérico. No início de cada iteração da linha 2, $\mathcal{L}$ é uma lista dos candidatos a i-ésimo caminho mínimo de s a t. Ao invés da partição Π de $\mathcal{P}$, Yen mantém em $\mathcal{L}$ um caminho mínimo de cada parte de Π. Em cada iteração é escolhido o caminho mais barato entre todos em $\mathcal{L}$ e a partição é *levemente* atualizada.

Método YEN (V, A, c, s, t, k)

1 $\quad \mathcal{L} \leftarrow \{$*um caminho de custo mínimo de* s *a* $t\}$
2 $\quad \mathcal{Q} \leftarrow \emptyset$
3 $\quad$ ***para*** $i = 1, \ldots, k$ ***faça***
4 $\quad\quad P_i \leftarrow$ *caminho de custo mínimo em* $\mathcal{L}$
5 $\quad\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{P_i\}$
6 $\quad\quad \mathcal{L} \leftarrow$ ATUALIZE $(V, A, c, \mathcal{Q})$
7 $\quad$ ***devolva*** $\langle P_1, \ldots, P_k \rangle$

Algoritmo ATUALIZE $(V, A, c, \mathcal{Q})$

0 $\quad \mathcal{L} \leftarrow \emptyset$
1 $\quad (N, E, f) \leftarrow$ *árvore dos prefixos de* $\mathcal{Q}$
2 $\quad$ ***para cada*** $u \in N$ ***faça***
3 $\quad\quad P_u \leftarrow$ *caminho de* s *a* t *de custo mínimo com prefixo* $f(R_u)$
e que não possui arcos em A_u
4 $\quad\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{P_u\}$
5 $\quad$ ***devolva*** $\mathcal{L}$

Na linha 3 do algoritmo ATUALIZE, na verdade, estamos resolvendo o problema SCM$(V, A, c, s, t, f(R_u), A_u)$. Assim, o consumo de tempo do algoritmo resultante é $n\, i\, T(n, m))$. Em chamadas consecutivas do algoritmo ATUALIZE, as árvores dos prefixos calculadas são muito semelhantes. De fato, o algoritmo pode ser implementado de tal maneira que o consumo o seu consumo de tempo seja $n\, T(n, m)$.

O método de YEN pode ser implementado de tal maneira que o seu consumo de tempo seja proporcional a $k\, n\, T(n, m)$.

7. Histórico e plano

Meu primeiro contato com o k-CM foi através do problema apresentado na introdução. Inicialmente, estudei o algoritmo de Naoki Katoh, Toshihide Ibaraki e H. Mine [8] (KIM) de uma maneira não-acadêmica, objetivando implementá-lo. Mais tarde, decidimos usar o k-CM e os algoritmos para resolvê-los como tema central do mestrado e passamos a estudar o algoritmo de KIM sobre um novo ponto de vista: entender sua essência, subrotinas, peculiaridades e, além disso, procurarmos semelhanças e diferenças com os demais algoritmos existentes para o mesmo problema.

O foco inicial do trabalho foi entender o algoritmo KIM para, num momento posterior, estudar a viabilidade de algumas mudanças experimentais que pudessem melhorar seu desempenho em grafos especiais ou, quem sabe, até em grafos genéricos. Durante a implementação do algoritmo KIM realizada na TeleMax, tive algumas idéias para melhorar o seu desempenho para o grafo que representava a rede de dados da TeleMax. Devido aos prazos curtos e, principalmente ao fato da implementação ter atendido aos requisitos de desempenho, não foi possível justificar orçamento para a análise e implementação das melhorias pensadas. Gostaria, nesta dissertação de mestrado, de apresentar a motivação do estudo do algoritmo KIM para o problema k-CM, estudá-lo à luz do método de Yen do qual ele é derivado, descrevê-lo de uma maneira mais simples, sem toda a especificidade de um pseudo-código, apresentar algumas melhorias, implementá-las e avaliar os seus desempenhos.

Até o momento estudamos o artigo de KIM, bastante adequado para aqueles que desejam apenas implementar o algoritmo, uma vez que o pseudo-código é apresentado em grande detalhe. A linguagem bastante carregada dificultou um entendimento do algoritmo em linhas gerais, razão que nos levou a buscar outra fonte. Embora não seja apenas um novo artigo sobre o algoritmo KIM, o trabalho de John Hershberger, Matthew Maxel e Subhash Suri [7] é classificado pelos autores como uma extensão do algoritmo KIM para grafos dirigidos. O grande mérito deste artigo, do nosso ponto de vista, não é o da apresentação de um novo algoritmo para o problema k-CM, mas sim pela descrição do problema k-CM e das idéias subjacentes na elaboração do algoritmo, apresentadas de uma maneira bem mais simples de ser compreendida, sem o abuso de notações pesadas, como as do artigo KIM.

Após algumas horas de estudo do artigo de Hershberger, Maxel e Suri, decidimos dar mais atenção ao método base para o problema k-CM, o método de Yen. Nosso objetivo era encontrar os fundamentos e as idéias mais gerais que permeavam, segundo nosso entendimento, todos os algoritmos para o problema k-CM.

A descrição do método de Yen [16] é bem sucinta, mas foi suficiente para entendermos algumas idéias. O plano agora é nos debruçarmos sobre o artigo de KIM, tendo como bagagem o aprendizado ganho dos trabalhos de Yen e de Hershberger, Maxel e Suri, lembrando que uma revisão destes artigos é recomendada. Após um entendimento melhor, passaremos à redação de um descrição mais "alto-nível" do algoritmo de KIM.

Pretendemos experimentar algumas mudanças no algoritmo KIM, e avaliar o quanto elas significam em ganho de desempenho. De antemão, sabemos que estas mudanças não acarretarão em melhoras assintóticas, mas acreditamos que conseguiremos alcançar desempenhos significativamente superiores. Como o algoritmo KIM tem como subrotina a geração de árvores de caminhos mínimos e, como foi constatado em Eleni Hadjiconstantinou e Nicos Christofides [6] que essa subrotina responde pela maior parte do processamento do algoritmo, estudaremos o algoritmo para a reconstrução de árvores de caminhos mínimos descrito por Enrico Nardelli, Guido Proietti e Peter Widmayer [13]. Resumidamente, se trata de um algoritmo para o seguinte problema: dada uma árvore de caminhos mínimos para um grafo G encontrar a árvore de caminhos mínimos para o grafo G' derivado de G pela

remoção de algumas arestas e vértices. Acreditamos que melhorias neste ponto do algoritmo possam levar a grandes ganhos de desempenho. O artigo de Alberto Marchetti-Spaccamela e Umberto Nanni [10] também está relacionado ao problema de reconstrução de árvores e faz parte da nossa agenda.

Vamos implementar uma versão bem crua dos algoritmos de Yen e KIM e algumas versões do algoritmo KIM usando as diversas alterações que pretendemos analisar. A idéia é construir uma versão para cada alteração proposta, outra com grupos de alterações escolhidas de modo a cobrir todos os casos. Em seguida, seguiremos com análises comparativas das versões utilizando diversos tipos de grafos gerados aleatoriamente.

Na tabela a seguir apresentamos, em linhas gerais, as diversas atividades para a execução do projeto, bem como uma expectativa de cumprimento destas atividades nos diversos meses do projeto.

Atividade	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ	JAN	FEV
Estudo de Yen[16]	√	√										
Estudo de Hershberger[7]			√	√								
Estudo de Katoh[8]		√	√									
Estudo de Nardelli[13]					√	√	√					
Implementações e comparativos				√	√	√	√	√	√	√	√	√
Redação da dissertação				√	√	√	√	√	√	√	√	√
Defesa da dissertação												√

Algumas atividades, como as implementações e a redação da dissertação, estarão presentes durante todo o projeto, razão pela qual aparecem na tabela anterior na maior parte dos meses.

Apesar de não pretendermos estudar o trabalho de David Eppstein [3], consideramos necessário, pelo menos, mencioná-lo neste ponto. Eppstein trata do problema de encontrar k-menores passeios, ou seja, pode haver repetição de vértices. Apesar desse problema parecer levemente diferente do k-CM, ele admite soluções muito mais eficientes.

Referências

[1] A.W. Brander e Mark C. Sinclair, A comparative study of k-shortest path algorithms, *11th UK Performance Engineering Workshop for Computer and Telecommunications Systems*, 1995, pp. 370–379.

[2] Edsger Wybe Dijkstra, A note on two problems in connection with graphs, *Numerische Mathematik* **1** (1959), 269–271.

[3] David Eppstein, Finding the k shortest paths, *SIAM Jounal on Computing* **28** (1998), no. 2, 652–673.

[4] Paulo Feofiloff, *Notas de aula de MAC5781 otimização combinatória*, `"http://www.ime.usp.br/~pf/"`, 1997.

[5] Michael L. Fredman e Robert Endre Tarjan, Fibonacci heap and their uses in improved network optimization algorithms, *Journal of the ACM* **34** (1987), 596–615.

[6] Eleni Hadjiconstantinou e Nicos Christofides, An efficient implementation of an algorithm for finding k shortest simple paths, *Networks* **34** (1999), no. 2, 88–101.

[7] John Hershberger, Matthew Maxel e Subhash Suri, Finding the k shortest simple paths: A new algorithm and its implementation, *ACM Transactions on Algorithms* **3** (2007), no. 4, 19 pages.

[8] Naoki Katoh, Toshihide Ibaraki e H. Mine, An efficient algorithm for k shortest simple paths, *Networks* **12** (1982), no. 4, 411–427.

[9] Eugene L. Lawler, A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem, *Management Science* **18** (1972), no. 7, 401–405.

[10] Alberto Marchetti-Spaccamela e Umberto Nanni, Fully dynamic algorithms for maintaining shortest path trees, *Journal of Algorithms* **34** (2000), 251–281.

[11] Ernesto Martins e Marta Pascoal, A new implementation of Yen's ranking loopless paths algorithm, *4OR Quart. J. Belgian, French, Italian Oper. Res. Soc.* **1** (2003), no. 2, 121–134.

[12] Ernesto Martins, Marta Pascoal e José Santos, *A new algorithm for ranking loopless paths*, Tech. report, Universidade de Coimbra, May 1997.

[13] Enrico Nardelli, Guido Proietti e Peter Widmayer, Swapping a falling edge of a single source shortest paths tree is good and fast, *Algorithmica* **35** (2003), 56–74.

[14] A. Perko, Implementation of algorithms for k shortest loopless paths, *Networks* **16** (1986), 149–160.

[15] Mikkel Thorup, Undirect single source shortest paths with positive integer weights in linear time, *Journal of the ACM* **46** (1999), 362–394.

[16] Jin Y. Yen, Finding the k shortest loopless paths in a network, *Management Science* **17** (1971), no. 11, 712–716.

Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508–900 São Paulo, SP

E-mail address: `pisaruk@ime.usp.br`