Problema do caminho mínimo

Estão descritos neste capítulo os ingredientes básicos que envolvem o problema do caminho mínimo, tais como função comprimento, função potencial, função predecessor, critério de otimalidade, etc. A referência básica para este capítulo é Feofiloff [13].

2.1 Descrição

Uma **função comprimento** em (V, A) é uma função de A em $\mathbb{Z}_{\geq}$. Se c é uma função comprimento em (V, A) e uv está em A, então, denotaremos por $c(u, v)$ o valor de c em uv. Se (V, A) é um grafo simétrico e c é uma função comprimento em (V, A), então c é **simétrica** se $c(u, v) = c(v, u)$ para todo arco uv. O **maior comprimento** de um arco será denotado por C, ou seja, $C = \max\{c(u, v) : uv \in A\}$. C

Se P é um passeio em um grafo (V, A) e c é uma função comprimento, denotaremos por $c(P)$ o **comprimento do caminho** P, ou seja, $c(P)$ é o somatório dos comprimentos de todos os arcos em P. Um passeio P tem **comprimento mínimo** se $c(P) \leq c(P')$ para todo passeio P' que tenha o mesmo início e término que P. A **distância** de um vértice s a um vértice t é o menor comprimento de um caminho de s a t. A distância de s a t em relação a c será denotada por $\text{dist}_c(s, t)$, ou simplesmente, quando a função comprimento estiver subentendida, $\text{dist}(s, t)$ denota a distância de s a t. $\text{dist}_c(s, t)$ $\text{dist}(s, t)$

Nesta dissertação, estamos interessados no seguinte **problema do caminho mínimo**:

> **Problema** $\text{PCM}(V, A, c, s)$: Dado um grafo (V, A), uma função comprimento c e um vértice s, encontrar um caminho de comprimento mínimo de s até t, para cada vértice t em V. PCM

Na literatura, essa versão do problema é conhecida como *single-source shortest path problem*.

2.2 Funções potencial e critério de otimalidade

Os algoritmos para o problema do caminho mínimo descritos nesta dissertação fornecem certificados de garantia para as suas respostas. Se um vértice t não é acessível a partir de s, um algoritmo pode, para comprovar este fato, devolver uma parte S de V tal que $s \in S$, $t \notin S$ e não existe uv com u em S e v em $V \setminus S$, i.e, $A(S) = \emptyset$. Este seria um certificado combinatório da não acessibilidade de t por s. Entretanto, os certificados fornecidos pelos algoritmos, baseados em funções potencial, serão um atestado compacto para certificar ambos, a minimalidade dos caminhos fornecidos, e a não acessibilidade de alguns vértices por s.

Uma **função potencial** é uma função de V em $\mathbb{Z}_{\geq}$. Se d é uma função potencial e c é uma função comprimento, então dizemos que d **respeita** c em uma parte A' de A se

$$d(v) - d(u) \leq c(u, v) \text{ para cada arco } uv \text{ em } A'.$$

Se d respeita c em A então diz-se que d é **viável**.

Funções potencial viáveis fornecem limitantes inferiores para comprimentos de caminhos. Suponha que c é uma função comprimento em (V, A) e que P é um caminho de um vértice s a um vértice t. Suponha ainda que d é uma função potencial viável. Vale que

$$c(P) \geq d(t) - d(s).$$

De fato, suponha que P é o caminho $\langle s = v_0, \alpha_1, v_1, \ldots, \alpha_k, v_k = t\rangle$. Tem-se que

$$\begin{aligned} c(P) &= c(\alpha_1) + \ldots + c(\alpha_k) \\ &\geq (d(v_1) - d(v_0)) + (d(v_2) - d(v_1)) + \ldots + (d(v_k) - d(v_{k-1})) \\ &= d(v_k) - d(v_0) = d(t) - d(s). \end{aligned}$$

Este fato é resumido através do lema a seguir, que é uma particularização do conhecido lema da dualidade de programação linear [14].

Lema 2.1 (lema da dualidade): *Seja (V, A) um grafo e c uma função comprimento sobre V. Para todo caminho P em (V, A) e toda função potencial viável d sobre V, vale que*

$$d(t) - d(s) \leq c(P),$$

onde s e t são o início e término de P, respectivamente. ∎

Do lema 2.1 tem-se imediatamente os seguintes corolários.

Corolário 2.2 (condição de inacessibilidade): *Se (V, A) é um grafo, c é uma função comprimento, d é uma função potencial viável e s e t são vértices tais que*

$$d(t) - d(s) \geq nC + 1$$

então t não é acessível a partir de s ∎

Corolário 2.3 (condição de otimalidade): *Seja (V, A) um grafo e c é uma função comprimento. Para toda função potencial viável e todo caminho P em (V, A) de s a t, se $d(t) - d(s) = c(P)$, então P é um caminho de comprimento mínimo.* ∎

2.3 Funções predecessor e representação de caminhos

Uma maneira compacta de representar caminhos de um dado vértice até cada um dos demais vértices de um grafo, é através de funções predecessor. Uma **função predecessor** é uma função de V em V. Se (V, A) é um grafo, ψ uma função predecessor sobre V e $v_0, v_1, \ldots, v_k$ são vértices tais que

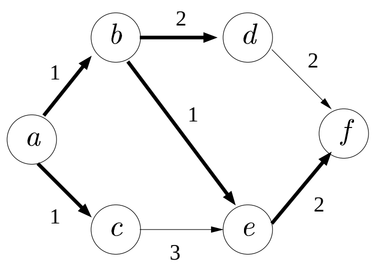
(1) $v_0 = \psi(v_1), v_2 = \psi(v_3), \ldots, v_{k-1} = \psi(v_k)$; e

(2) $\alpha_i = v_{i-1}v_i$ está em A para $i = 1, \ldots, k$,

então dizemos que $\langle v_0, \alpha_1, v_1, \ldots, \alpha_k, v_k\rangle$ é um **caminho determinado por ψ**.

Seja ψ uma função predecessor e $\Psi := \{uv \in A : u = \psi(v)\}$. É dito que ψ **determina uma arborescência** quando o grafo (V, Ψ) é uma arborescência.

Os algoritmos descritos nesta dissertação utilizam funções predecessor para, compactamente, representar todos os caminhos de comprimento mínimo a partir de um dado vértice, conforme ilustrado na figura 2.1.



vértice	ψ
a	a
b	a
c	a
d	b
e	b
f	e

Figura 2.1: Representação de caminhos através da função predecessor ψ com vértice inicial a. Os números próximos aos arcos representam a função comprimento. Os arcos em destaque formam uma arborescência. A tabela ao lado mostra os valores de ψ.

2.4 Examinando arcos e vértices

Algoritmos para encontrar caminhos mínimos mantém, tipicamente, além de uma função predecessor, uma função potencial. O valor desta função potencial para cada vértice é um limitante superior para a distância a partir do vértice s. Esta função é intuitivamente interpretada como uma **distância tentativa** a partir de s.

Seja d uma função potencial e ψ uma função predecessor. Uma operação básica envolvendo as funções ψ e d é **examinar um arco** (*relaxing* [9], *labeling step* [38]). Examinar um arco uv consiste em verificar se d respeita c em uv e, caso não respeite, ou seja, caso

$$d(v) - d(u) > c(u,v) \quad \text{ou, equivalentemente} \quad d(v) > d(u) + c(u,v),$$

fazer

$$d(v) := d(u) + c(u,v) \quad \text{e} \quad \psi(v) := u.$$

Intuitivamente, ao examinar um arco uv tenta-se encontrar um "atalho" para o caminho de s a v determinado por ψ, passando por uv. O passo de examinar uv pode diminuir o valor da distância tentativa dos vértices v e atualizar o predecessor, também tentativo, de v no caminho de comprimento mínimo de s a v. O consumo de tempo para examinar um arco é constante.

Outra operação básica é **examinar um vértice**. Se u é um vértice, examinar u consiste em examinar todos os arcos da forma uv. Em linguagem algorítmica tem-se

Para cada arco uv faça
  se $d(v) > d(u) + c(u,v)$
    então $d(v) := d(u) + c(u,v)$ e $\psi(v) := u$.

O consumo de tempo para examinar um vértice é proporcional ao número de arcos com ponta inicial no vértice.

2.5 Complexidade

O problema do caminho mínimo (PCM), na sua forma mais geral, ou seja, aceitando comprimentos negativos, é NP-difícil: o problema do caminho hamiltoniano pode facilmente ser reduzido a este problema.

Existem limitantes inferiores bem "fracos" para a complexidade do PCM no modelo comparação-adição. Spira e Pan [37] mostraram que, independente do número de adições, pelo menos cn^2 comparações são necessárias para resolver o problema em um grafo simétrico completo[1] com

[1] Um grafo simétrico (V, A) é **completo**, se uv e vu estão em A para todo par de vértices distintos u e v.

n vértices, onde c é uma constante positiva. Limitantes para outros problemas relacionados são listados em Pettie e Ramachandran [31].

Alguns algoritmos para o PCM utilizam como subrotina um algoritmo para o **problema da árvore geradora mínima**, que consiste em

> **Problema** AGM(V, A, c): Dado um grafo simétrico (V, A), uma função comprimento simétrica c, encontrar uma árvore geradora de comprimento mínimo. AGM

O comprimento de uma árvore é a soma dos comprimentos das suas arestas.

Pettie e Ramachandran [31] observaram que, no pior caso, o tempo necessário para resolver o problema da árvore geradora mínima, no modelo de comparação, não é superior ao tempo necessário para resolver o problema do caminho mínimo no modelo de comparação-adição. Desta forma, um algoritmo para o problema do caminho mínimo, pode utilizar uma subrotina que constrói uma árvore geradora mínima sem que isto comprometa o seu desempenho assintótico no pior caso.

No modelo RAM, o problema AGM pode ser resolvido em tempo linear, como descrito por Fredman e Willard [18]. Assim, um algoritmo para o PCM, no modelo RAM, pode utilizar, digamos, um algoritmo para o problema AGM como pré-processamento, sem comprometer a eficiência teórica do algoritmo. O algoritmo de Thorup, descrito no capítulo 6, resolve o problema AGM em seu pré-processamento.