4
Algoritmos para o MBH

Descrevemos neste capítulo os algoritmo para o MBH propostos por [14] e [12]. Além disso, fornecemos um modelo em programação inteira para o MBH. Finalmente, mostramos como adaptar o algoritmo PATH para o MBH.

4.1
Algoritmo GREEDY-BFS

O algoritmo GREEDY-BFS foi projetado por Czyzowicz et al. [14]. Este algoritmo é guloso por *ganho*, onde o ganho de um hotlink $(u, v) \in A$ em uma árvore T^A é definido como

$$g(u, v, T^A) = p(v, T^A)\ (d(v) - d(u) - 1), \tag{4-1}$$

onde $p(v, T)$ é a soma das probabilidades das hotleaves descendentes de v em T^A, e $d(v)$ é o nível de v na árvore sem os hotlinks T. Observe que se o hotlink (u, v) pertence a uma atribuição de hotlinks viável A, e alguma hotleaf é acessível partindo de v, então $g(u, v, T^A)$ representa a redução no custo $E[T^A, \mathbf{p}]$ provocada por este hotlink. Isto é verdade pelo fato de assumirmos a "navegação óbvia". O pseudo-código do algoritmo GREEDY-BFS é fornecido na Figura 4.1.

4.2
Algoritmo KRANAKIS

O algoritmo KRANAKIS foi proposto por Kranakis et al [12]. A única diferença entre este algoritmo e o algoritmo GREEDY-BFS é o critério usado para escolher o nó v. KRANAKIS escolhe um nó v tal que $\frac{p(u,T^A)}{d+1} \leq p(v, T^A) \leq \frac{d.p(u,T^A)}{d+1}$. Se este nó não existe, ou se é um filho de u, então o algoritmo escolhe um neto de u com maior probabilidade. Esta

Algoritmo 6: GREEDY-BFS
Entrada: T: árvore direcionada
Saída: A: atribuição de hotlinks
(1) $A \leftarrow \emptyset$
(2) **foreach** nó u em uma busca em largura em T
(3) Seja v um nó descendente de u em T^A que maximiza o ganho $g(u, v, T^A)$
(4) **if** v é definido **then** $A \leftarrow A \cup \{(u, v)\}$
(5) **return** A

Figura 4.1: Pseudo-código do algoritmo GREEDY-BFS.

estratégia permite provar uma aproximação que depende do grau máximo d. Apresentamos o pseudo-código deste algoritmo na Figura 4.2.

Algoritmo 7: KRANAKIS
Entrada: T: árvore direcionada
Saída: A: atribuição de hotlinks
(1) $A \leftarrow \emptyset$
(2) **foreach** nó u em uma busca em largura em T
(3) Seja v um nó descendente de u em T^A tal que $\frac{p(u,T^A)}{d+1} \leq p(v, T^A) \leq \frac{d.p(u,T^A)}{d+1}$
(4) **if** v não é definido **or** $v \in S(u, T^A)$
(5) Seja v um neto de u em T^A que maximiza $p(v)$
(6) **if** v é definido **then** $A \leftarrow A \cup \{(u, v)\}$
(7) **return** A

Figura 4.2: Pseudo-código do algoritmo KRANAKIS.

4.3 Modelo em Programação Inteira

Para modelar o MBH como um problema de programação inteira, consideramos os m caminhos $(\lambda_1, \ldots, \lambda_m)$ na árvore de entrada T, partindo da raiz até cada uma das m hotleaves. Denotamos por c_k e p_k o comprimento do caminho λ_k e a probabilidade da única hotleaf neste caminho, respectivamente. Denotamos também por $\mathcal{A}$ o conjunto dos hotlinks que

estaremos decidindo adicionar em T, ou seja, $\mathcal{A} = \{(u,v) \in (V \times V) - E \mid v \text{ é descendente de } u\}$.

Em nosso modelo, utilizamos a variável binária $x_{i,j}$ para indicar se o hotlink (i,j) pertence a solução ($x_{i,j} = 1$) ou não ($x_{i,j} = 0$). Utilizamos também uma variável contínua $y^k_{i,j}$ para representar a redução no comprimento do caminho λ_k devido a adição do hotlink (i,j). Assim, depois da atribuição dos hotlinks, o comprimento do caminho λ_k é dado pela expressão $c_k - \sum_{i,j \in \lambda_k} y^k_{i,j}$.

Dizemos que dois hotlinks (i,j) e (a,b) são *aninhados* quando existe pelo menos um caminho que contêm i,j,a,b, com $(i \neq a) \vee (b \neq j)$, e $d(i) \leq d(a) < d(b) \leq d(j)$. Ilustramos hotlink aninhados na Figura 4.3.(a).

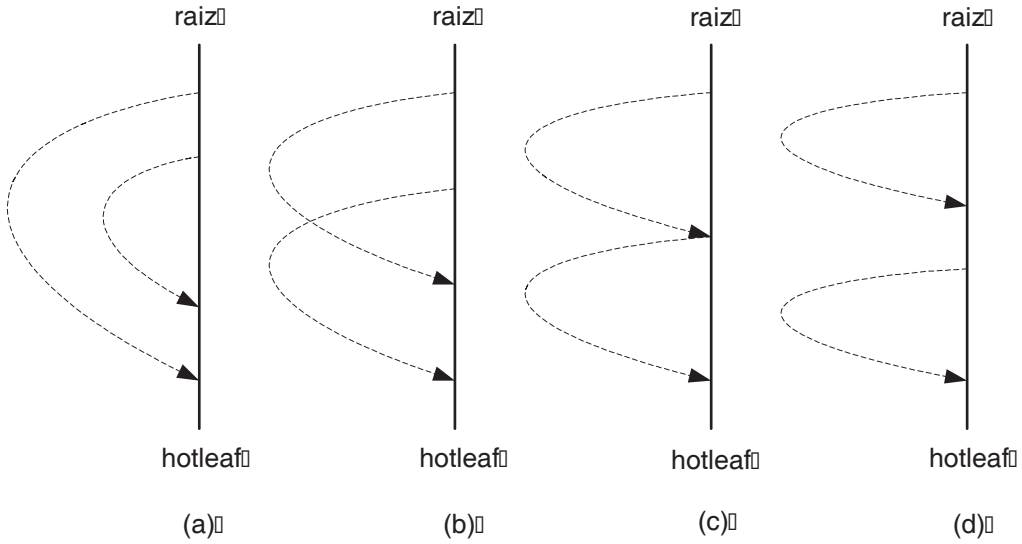

Figura 4.3: As quatro configurações possíveis para um par de hotlinks em um caminho. Os hotlinks em (a) são "aninhados". Os hotlinks em (b) são "cruzados".

Se dois hotlinks são aninhados, então apenas a redução no comprimento do caminho causada pelo hotlink mais externo deve ser não nula. Além disso, qualquer redução de comprimento de caminho causada pela adição do hotlink (i,j) não é maior que $d(i,j) = d(j) - d(i) - 1$. Portanto, temos as seguintes restrições em nosso modelo:

$$y^k_{a,b} \leq (1 - x_{i,j})\ d(a,b), \quad \forall (i,j),(a,b) \in \mathcal{A} \text{ aninhados};$$
$$y^k_{a,b} \leq d(a,b)\ x_{a,b}, \quad \forall (a,b).$$

Dizemos que dois hotlinks (i,j) e (a,b) "cruzam"quando existe pelo menos um caminho que contêm i,j,a,b, e $d(i) < d(a) < d(j) < d(b)$. Ilus-

tramos hotlinks cruzados na Figura 4.3.(b). Com a suposição da navegação óbvia, temos que um dos hotlinks cruzados nunca é utilizado pelos usuários. Portanto, adicionamos a seguinte restrição:

$$x_{i,j} + x_{a,b} \leq 1, \quad \forall (i,j),(a,b) \in \mathcal{A} \text{ cruzados.}$$

Finalmente, como podemos usar no máximo um hotlink por nó, temos que

$$\sum_{j=1}^{n} x_{i,j} \leq 1, \quad \forall i \in V.$$

A contribuição do caminho λ_k no custo $E[T^A, \mathbf{p}]$ é dado pelo comprimento de λ_k em T^A, multiplicado por p_k. Logo, o objetivo do MBH é modelado como

$$\min \sum_{k=1}^{m} p_k \left(c_k - \sum_{i,j \in \lambda_k} y_{i,j}^k \right).$$

4.4 Algoritmo M-PATH

Chamamos de M-PATH o algoritmo PATH adaptado para resolver o MBH. As modificações necessárias no algoritmo PATH são as seguintes:

(i) substituímos $\max\{h^*(\mathbf{q} \rightarrow r, \mathbf{c}(1,1), \mathcal{T}_f), h^*(\mathbf{q}, \overline{\mathbf{c}}(0), \mathcal{T} - \mathcal{T}_f)\}$ na expressão (3-5) por $h^*(\mathbf{q} \rightarrow r, \mathbf{c}(1,1), \mathcal{T}_f) + h^*(\mathbf{q}, \overline{\mathbf{c}}(0), \mathcal{T} - \mathcal{T}_f)$;

(ii) a condição de parada dada pela expressão (3-6) deve ser modificada para

$$h^*(\mathbf{q}, \mathbf{a}, \mathcal{T}) = \begin{cases} \min\{i | 1 \leq i \leq k \text{ e } a_i = 1\} p_l, & \text{se } \mathbf{q} \text{ tem algum nó disponível} \\ p_l k, & \text{caso contrário} \end{cases}$$

onde p_l é a probabilidade da folha l em $\mathcal{T}$.

No MBH, o custo de um par de sub-problemas gerados pela decomposição do caso 2 é a soma dos custos dos sub-problemas. Utilizamos o máximo destes custos no PBH. Portanto, a modificação (i) é necessária. Como o custo de um caminho no MBH depende também da probabilidade da folha neste caminho, devemos aplicar a modificação (ii).

Este algoritmo tem as mesmas complexidades do algoritmo PATH, como indica o Teorema 4.1 abaixo.

Teorema 4.1 *O algoritmo M-PATH executa em tempo* $O(n3^D)$ *e usa* $O(n2^D)$ *de espaço.*

Prova. A prova é análoga a do Teorema 3.8. □