1
Introdução

1.1
Motivação

Encontrar a informação desejada em uma base de dados grande e caótica como a Internet é uma tarefa complexa. Basicamente, esta busca pode ser feita de duas maneiras. Uma delas é usar uma linguagem de consulta, como fazem os sites de busca. A outra é utilizar um índice hierárquico que respeita alguma maneira de categorizar os dados.

Exemplos desta hierarquia são os serviços de diretório, como em [1]. Na Figura 1.1 temos uma página do diretório do Yahoo que representa a categoria "Computer Science", que é uma sub-categoria da categoria "Science". O usuário deverá passar por esta página se deseja encontrar endereços de sites que falam sobre "Sorting Algorithms", como vemos na Figura 1.2.

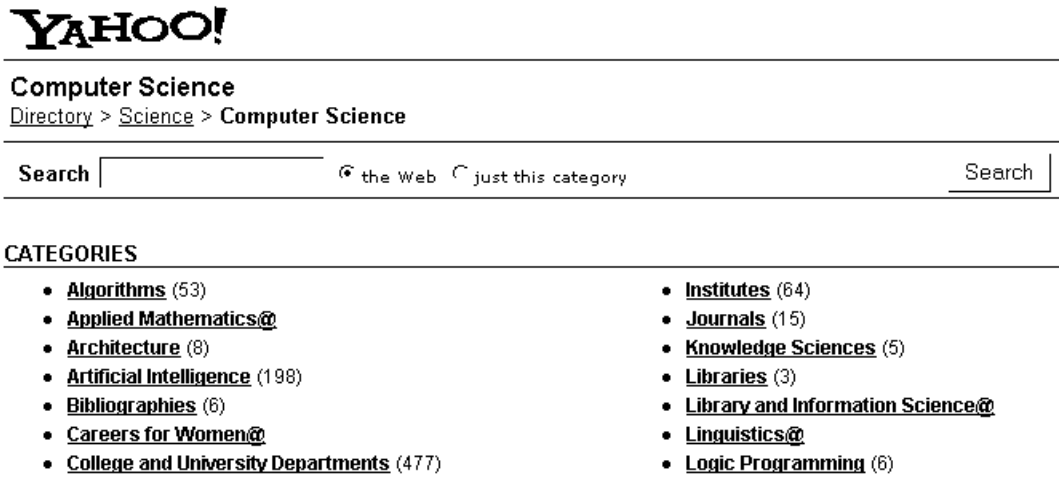

Figura 1.1: Página web que corresponde a um nó interno na árvore que representa o diretório do Yahoo.

O uso de índice hierárquico tem algumas desvantagens. O número de operações necessárias para encontrar a informação é muito maior que o uso

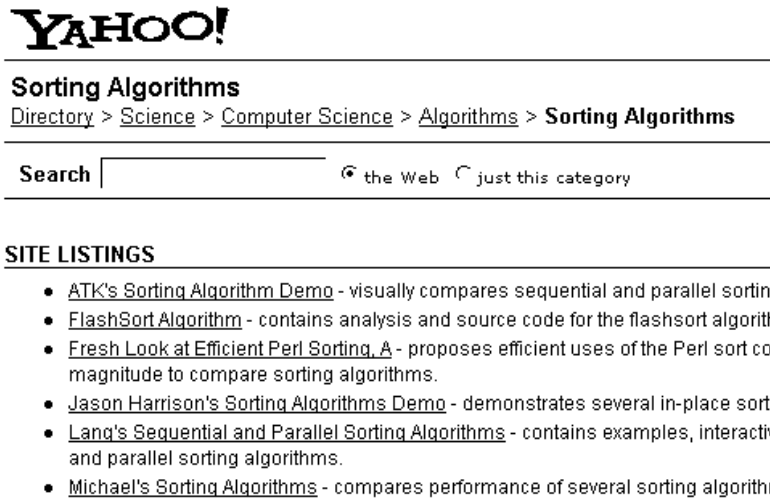

Figura 1.2: Página web que corresponde a um nó folha na árvore que representa o diretório do Yahoo.

de uma consulta correta. É necessário percorrer a árvore de índices da raiz até o nó folha desejado. Além disso, como é difícil para um humano escolher um item em uma lista muito longa, o grau da árvore deve ser mantido baixo[1]. Isto agrava o problema, pois aumenta a profundidade média.

Outro problema da abordagem hierárquica é que a profundidade de um item na árvore não está relacionada com o padrão de acesso dos usuários. Com isso, um item que tem alta frequência de acesso pode requerer um caminho muito longo, enquanto um item pouco "popular" pode ser facilmente acessado na árvore.

Desejamos reduzir o número de acessos que um usuário faz numa busca sem modificar a árvore de índices, pois esta categorização auxilia na procura pela informação desejada. A abordagem que consideramos aqui é o uso de "hotlinks". Hotlinks são hyperlinks adicionais que servem como "atalhos" numa busca. Como não podemos inserir muitos hotlink em um nó, tratamos o problema de atribuir no máximo um hotlink em cada nó da árvore. Formalizamos este problema na seção seguinte. Muitas idéias apresentadas aqui podem ser extendidas para o caso de k hotlinks por nó.

1.2 Definição do problema

Podemos modelar um site web como um grafo direcionado $G = (V, E)$, onde cada nó $u \in V$ representa uma página do site, e cada arco $(u, v) \in E$ indica que existe um hyperlink na página representada por u para a página

[1] Um número conveniente é entre 7 e 10.

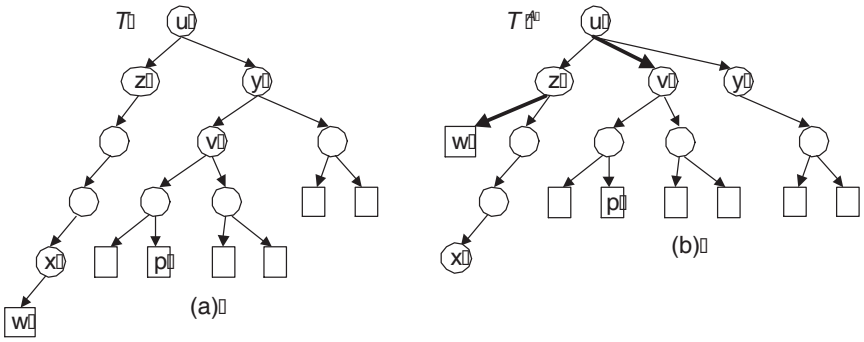

Figura 1.3: (a) a árvore T. (b) a árvore T^A, onde $A = \{(u, v), (z, w)\}$.

representada por v. Estamos interessados em sites que podem ser modelados como uma árvore direcionada enraizada, como na Figura 1.3.(a).

É importante também estabelecer em nosso modelo como os usuários navegam no site. Assumimos que toda informação que os usuários procuram estão nos nós folha (quadrados na Figura 1.3.(a)). Cada folha tem uma probabilidade de ser acessada, que corresponde a *popularidade* da página. Além disso, os usuários sempre fazem uma busca por alguma informação partindo do nó raiz, que corresponde a home page do site. Portanto, os nós internos são utilizados apenas para navegação, permitindo que o usuário encontre a informação desejada dentre uma grande quantidade de páginas.

Assumimos também que um usuário *sempre* sabe qual link tomar, apesas de não dispor de um mapa do site. Os usuário decidem com base na categorização hierárquica que os nós internos representam. Na prática, um usuário pode errar o caminho na busca, mas não consideramos este caso em nosso modelo.

Quando um site possui hotlinks, pode haver mais de uma opção de link para tomar em um nó. Neste caso, assumimos que o usuário sempre toma um hotlink (u, v) quando, partindo de u, busca uma folha na sub-árvore com raiz em v. Esta consideração é chamada de "navegação óbvia", e foi proposta em [14]. Apesar de não ser uma estratégia ótima, é a melhor opção para um usuário que não possui um mapa do site.

Como consequência da "navegação óbvia", temos que a inclusão de um hotlink (u, v) deve ser seguida da remoção[2] de qualquer outro arco que chega em v, pois este outro arco nunca é utilizado. Portanto, o grafo obtido após inserir um hotlink em um árvore é também uma árvore. Por exemplo, a árvore da Figura 1.3.(b) é obtida da árvore da Figura 1.3.(a) através da adição dos hotlinks (u, v) e (z, w). Vamos agora definir uma *atribuição de*

[2]Removemos este arco apenas da representação do site, não do site propriamente dito, pois devemos conservar sua estrutura original.

hotlinks viável.

Lembramos algumas definições básicas. O nível de um nó em uma árvore é o número de arcos no caminho que conecta este nó à raiz da árvore. Dizemos que um nó v é descendente de outro nó u em T quando o único caminho em T que conecta r a v contêm u. Neste caso, temos também que u é ancestral de v. Um nó u é um descendente (ancestral) próprio de v se u é um descendente (ancestral) de v e $u \neq v$.

Dada $T = (V, E)$, uma *atribuição de hotlinks* é definida como um conjunto $A \subset V \times V$. Uma atribuição de hotlinks é *viável* se e somente se satisfaz as seguintes condições:

(i) para todo arco $(u, v) \in A$, v é descendente de u em T;

(ii) sejam u, v, a, b nós em V tal que u é um ancestral próprio de v e v é um ancestral próprio de a. Se $(u, a), (v, b) \in A$, então a não é um ancestral de b;

(iii) para todo nó $u \in V$, existe no máximo um arco $(u, v) \in A$.

Como um hotlink é um "atalho", ele deve sempre apontar para um descendente, conforme estabelece a condição (i). A condição (ii) está relacionada com a "navegação óbvia", e sua motivação pode melhor compreendida examinando a Figura 1.3. A condição previne, por exemplo, que (u, v) e (y, p) pertençam a A simultaneamente. Lembre que assumimos que o usuário sempre toma o hotlink (u, v) quando busca uma folha na sub-árvore com raiz em v, partindo de u. Como p é um nó nesta sub-árvore, concluímos que o hotlink (y, b) (se existir) nunca será seguido pelo usuário. Finalmente, a condição (iii) limita em no máximo um o número de hotlinks que podemos utilizar em um nó.

Note que a definição de atribuição de hotlinks viável permite laços. Embora não sejam necessários, eles ajudam na descrição dos algoritmos propostos. Na prática, porém, nunca adicionamos laços.

Podemos agora discutir como melhorar os sites com o uso de hotlinks. Trabalhamos em duas melhorias possíveis devido ao uso de hotlinks: redução do número de acessos em uma busca, no pior caso ou no caso médio. Chamamos o problema de minimizar o número máximo de acessos de *Pior Caso de Busca com Hotlinks* (PBH). Por outro lado, chamamos o problema de minimizar o número médio de acessos de *Caso Médio de Busca com Hotlinks* (MBH). Uma instância do PBH é definida por uma árvore direcionada $T = (V, E)$ com raiz no nó $r \in V$. Para o MBH, temos também uma distribuição de probabilidades $\mathbf{p}$ associada às folhas de T, onde p_x é

a probabilidade de acesso da folha x. Assim, uma instância para o MBH é definida pelo par $(T, \mathbf{p})$.

Agora formalizamos os objetivos dos problemas PBH e MBH. Para isso, introduzimos o conceito de *árvore melhorada*.

Definição 1.1 *Dada $T = (V, E)$, e uma atribuição de hotlinks A, a árvore melhorada obtida de T após a atribuição A é definida como $T^A = (V, (E - X) \cup A)$, onde $X = \{(u, v) \in E \mid (y, v) \in A$ para algum $y \in V\}$.*

Na definição acima, X é o conjunto de arcos de E cujos nós de origem são apontados por hotlinks em A. Por exemplo, a Figura 1.3.(a) mostra um árvore T e a Figura 1.3.(b) mostra a árvore $T^{\{(u,v),(z,w)\}}$. O conjunto X neste caso é $\{(x, w), (y, v)\}$.

Dada uma árvore melhorada T^A com raiz em r, seja $d_A(u)$ o nível de u em T^A. Além disso, seja L o subconjunto de V que contêm todas as folhas de T. Uma atribuição de hotlinks M-ótima (onde M significa caso médio) para $(T, \mathbf{p})$ é uma atribuição de hotlinks viável A^* que minimiza

$$E[T^{A^*}, \mathbf{p}] = \sum_{x \in L} p_x d_{A^*}(x)$$

dentre todas as atribuições viáveis possíveis. Por outro lado, uma atribuição de hotlinks P-ótima (onde P significa pior caso) para T é uma atribuição de hotlinks viável A^* que minimiza

$$h(T^{A^*}) = \max_{x \in L} d_{A^*}(x),$$

dentre todas as atribuições viáveis possíveis. O objetivo do problema MBH (PBH) é encontrar uma atribuição de hotlinks M-ótima (P-ótima) para T.

Finalmente, estabelecemos a seguir notações que utilizamos ao longo do texto.

Notação

Usamos $h^*(T)$ e $E^*[T, \mathbf{p}]$, respectivamente, para denotar $h(T^{A^*})$ e $E[T^{A^*}, \mathbf{p}]$. Observe que alguns nós internos de T podem se tornar folhas em T^A (por exemplo, o nó x na Figura 1.3.(b)). Por definição, estes nós não pertencem a L. Portanto, nos referimos a estes nós de L como *hotleaves*. A *altura* da árvore T', denotada por $H(T')$, é a máxima distância da raiz de T' para uma folha em T'. A altura leva em conta todas as folhas, inclusive aquelas que não são hotleaves. Na Figura 1.3.(b), temos

$H(T^A) = 4$ enquanto $h(T^A) = 3$. Esta medida tem um papel importante na complexidade do nosso algoritmo exato para o PBH. Utilizaremos H para denotar a altura $H(T)$ da árvore de entrada T.

Usamos também $T_u = (V_u, E_u)$ para denotar o sub-grafo de T induzido pelos descendentes de u, ou seja, $V_u = \{v \in V \mid v \text{ é descendente de } u\}$, e $E_u = \{(v, w) \in E \mid v, w \in V_u\}$. Nos referimos a T_u como *sub-árvore máxima de T com raiz em u*. $T - T_u$ é usado para denotar o sub-grafo de T induzido por $V - V_u$. Uma sub-árvore binária de T é uma sub-árvore de T com grau máximo de saída 2.

Além disso, usamos $S(u, T)$ para denotar o conjunto de todos os filhos do nó u em T. Se $u \in L$ então dizemos que T_u é uma *sub-árvore trivial* de T. Finalmente, usamos $\log x$ para denotar $\log_2 x$, e d como o grau máximo de saída dos nós em T.

1.3 Trabalhos relacionados

Devido à importância e ao rápido crescimento da Internet, um esforço considerável tem sido feito no sentido de melhorar sua performance. Duas abordagens importantes são "entender a topologia"[8, 9, 10] e "design adaptativo"[7, 6]. O uso de hotlinks é uma proposta intermediária e foi sugerida por Perkowitz and Etzioni [7]. Depois disso, surgiram vários estudos relacionados ao uso de hotlinks em sites. Vejamos alguns destes trabalhos.

Em [11], Bose et. al trabalharam com o problema MBH sem a consideração de "navegação óbvia", ou seja, o usuário toma o melhor link, podendo ou não ser um hotlink. Eles provaram limites superiores e inferiores para o caso de árvores binárias completas com distribuições de probabilidades uniforme, geométrica, zipf, e com distribuições arbitrárias. O limite inferior para distribuições arbitrárias pode ser extendido para árvores de grau máximo d. Como este limite vale também para o caso em que a "navegação óbvia"é considerada, temos um limite inferior de $\frac{Ent(\mathbf{p})}{\log(d+1)}$ para $E^*[T, \mathbf{p}]$, onde $Ent(\mathbf{p})$ é a entropia [3] da distribuição de probabilidades $\mathbf{p}$. Eles consideraram também uma variação onde a árvore de entrada é substituída por um grafo direcionado acíclico arbitrário. Neste caso, eles mostraram que o problema é $\mathcal{NP}$-completo, mesmo com distribuição uniforme.

Kranakis et al. [12] apresentaram um algoritmo aproximado (chamamos de KRANAKIS) de tempo $O(n^2)$, cuja análise fornece o

seguinte resultado:

$$E^*[T, \mathbf{p}] \leq \frac{Ent(\mathbf{p})}{\log(d+1) - (d/(d+1))\log d} + \frac{d+1}{d}$$

Eles também mostraram que o algoritmo obtêm soluções próximas do ótimo para árvores completas k-regulares.

Vargas et. al [14] desenvolveram uma heurística (chamada GREEDY-BFS) para o MBH e realizaram experimentos com sites reais, comparando esta heurística com o algoritmo aproximado proposto em [12]. A heurística GREEDY-BFS e o algoritmo aproximado KRANAKIS são descritos no Capítulo 4. Utilizamos estes algoritmos em nossos experimentos, descritos no Capítulo 5 e em [17].

Em [13], Fuhrmann et al. consideraram a versão do MBH onde múltiplos hotlinks podem ser atribuídos em cada nó. Aqui também os autores não consideraram a "navegação óbvia". Para árvores completas k-regulares, eles provaram limites superiores para $E^*[T, \mathbf{p}]$ com distribuições arbitrárias. Provaram também limites inferiores para $E^*[T, \mathbf{p}]$ no caso de distribuição uniforme.

Gerstel et al. [15] apresentaram um algoritmo exato para o MBH baseado em programação dinâmica. Este algoritmo é polinomial para árvores de altura logarítmica. O mesmo resultado foi obtido independentemente por Pessoa et al. [16]. Por outro lado, para o PBH, Pessoa et al. apresentam um algoritmo polinomial para qualquer árvore de entrada, enquanto Gerstel et al. fornecem um algoritmo de aproximação constante. Gerstel et al. mostraram que a prova de $\mathcal{NP}$-completude realizada em [11] pode ser facilmente adaptada para provar que o problema MBH, com grafos direcionados acíclicos arbitrários, também é $\mathcal{NP}$-completo quando consideramos a "navegação óbvia".

1.3.1 Contribuições

As contribuições para os problemas PBH e MBH descritas nesta dissertação são mais concisamente apresentadas em [16, 17]. Damos a seguir uma visão geral destes resultados.

Para o problema PBH, apresentamos um algoritmo (14/3)-aproximado que executa em tempo $O(n \log m)$ e requer espaço linear, onde $n = |V|$ e $m = |L|$. Uma idéia simples de algoritmo aproximado seria procurar um nó u tal que a adição do hotlink (r, u) gera sub-problemas com aproximadamente

o mesmo número de hotleaves. Embora esta idéia seja útil para provar um limite superior para $h(T^{A^*})$, ela não fornece um algoritmo aproximado de fator constante, como veremos no Capítulo 2. Para conseguir um fator constante, escolhemos um nó u tal que o hotlink (r, u) gera sub-problemas "balanceados", onde o critério de balanceamento é determinado por um limite inferior para $h^*(T)$. Descrevemos este algoritmo na Seção 3.1.

Introduzimos um algoritmo exato para o PBH, baseado em programação dinâmica, que executa em tempo $O(n3^D)$ e usa $O(n2^D)$ de espaço, onde D é um limite superior para a altura de T^{A^*}. Chamamos este algoritmo de PATH. Como provamos no Capítulo 2 que $D = O(\log n)$ para o PBH, temos que o algoritmo exato para este problema é polinomial ($O(n(nm)^{2.284})$ de tempo e $O(n(nm)^{1.441})$ de espaço). É importante mencionar que a aplicação direta da técnica de programação dinâmica ao problema PBH produz um algoritmo exponencial no tamanho da árvore de entrada T.

O algoritmo exato para o PBH pode facilmente ser adaptado para resolver o MBH. Chamamos este algoritmo adaptado de M-PATH. Entretanto, para este problema não conhecemos um limite logarítmico para H. Portanto, este algoritmo é polinomial para o MBH apenas para instâncias onde o H é logarítmico em n (por exemplo, árvores completas k-regulares). Como o algoritmo é exponencial, criamos um parâmetro D que permite melhorar o tempo de execução em detrimento da qualidade da solução. A redução na qualidade da solução é feita limitando a altura da árvore melhorada, ou seja, nosso algoritmo produz a melhor atribuição de hotlinks possível A^*_D dentre as atribuições de hotlinks A tal que $H(T^A) \leq D$. Este algoritmo executa em tempo $O(n3^D)$.

Finalmente, realizamos experimentos com alguns algoritmos disponíveis para o MBH. O objetivo foi comparar a performance e a qualidade das soluções obtidas. Comparamos os algoritmos: M-PATH, GREEDY-BFS, KRANAKIS e um modelo em programação inteira implementado no XPRESS [2]. Introduzimos melhorias práticas que fizeram nosso algoritmo ter boa performance nos sites reais utilizados. Além disso, o algoritmo M-PATH obteve a solução ótima de praticamente todas as instâncias. Descrevemos os experimentos no Capítulo 5.

1.4 Organização

No capítulo 2 provamos propriedades estruturais para o PBH e MBH. Os limites inferiores e superiores demonstrados neste capítulo são fundamentais para o entendimento do algoritmo aproximado, e para a complexidade polinomial do algoritmo PATH. Estes algoritmos são descritos no Capítulo 3, onde falamos dos algoritmos propostos para o PBH. Os algoritmos para o MBH são apresentados no Capítulo 4. Neste capítulo damos mais detalhes dos algoritmos GREEDY-BFS e KRANAKIS, além de mostrar como adaptamos o algoritmo PATH para o MBH. Apresentamos também neste capítulo um modelo em programação inteira para o MBH. Em seguida, descrevemos nossos experimentos no Capítulo 5. Finalmente, damos as conclusões e propostas de trabalhos futuros no Capítulo 6.