5
Experimentos

Neste capítulo, descrevemos a avaliação experimental dos algoritmos para o MBH apresentados no Capítulo 4. Na Seção 5.1 discutimos o ambiente experimental e a construção das instâncias utilizadas. Explicamos na Seção 5.2 idéias utilizadas na implementação do algoritmo M-PATH para melhorar sua performance e consumo de memória. Finalmente, apresentamos os resultados experimentais na Seção 5.3.

5.1
Ambiente e instâncias

Os experimentos foram executados no ambiente Linux em uma máquina Xeon 2.4 GHz com 1GB de memória RAM. O modelo em programação inteira foi implementado no pacote de otimização XPRESS-MP [2].

Para obter nossas instâncias, adotamos a mesma abordagem empregada por Czyzowicz [14]. Inicialmente obtemos os grafos direcionados associados aos sites de 84 universidades brasileiras e americanas. Sejam $G^1, \ldots, G^{84}$ estes grafos, e seja s_i, para $i = 1, \ldots, 84$, o nó que modela a home-page do i-ésimo site. Em seguida, para $i = 1, \ldots, 84$, executamos uma busca em largura em G^i, partindo de s_i, para extrair a árvore T^i. Este processo gerou 84 árvores.

A Tabela 5.1 indica alguns parâmetros das árvores iniciais.

Para atribuir probabilidades às folhas, empregamos a distribuição de Zipf, onde a probabilidade da i-ésima folha mais provável é dada por $p_i = \frac{1}{iH_m}$. Aqui, H_m é o número harmônico $H_m = \sum_{j=1}^{m} \frac{1}{j}$. O uso desta distribuição é motivado pelo trabalho de Glassman [5], que prova experimentalmente que a popularidade de páginas web podem ser modeladas com esta distribuição de probabilidades. A ordem de popularidade entre as folhas foi gerada aleatoriamente. Note que o tempo de execução de M-PATH para um dado valor de D não depende da distribuição de probabilidades uti-

	Site	n	m	d	H	$E[T, \mathbf{p}]$
1	www.ufs.br	542	440	47	6	3.614161
2	www.ufrn.br	320	254	28	7	5.065515
3	www.pucminas.br	1016	921	81	4	3.008704
4	www.ufmg.br	2275	1863	153	6	4.796833
5	www.pucsp.br	1151	938	43	7	5.132679
6	www.ucb.br	1158	933	52	7	5.418895
7	www.ucsal.br	369	219	43	5	3.973058
8	www.ufg.br	556	477	78	4	2.740458
9	www.ucg.br	2317	2016	76	6	4.477387
10	www.ufmt.br	746	583	84	9	5.623212
11	www.ufms.br	1743	1534	390	3	2.613837
12	www.ufpr.br	646	554	80	5	3.530420
13	www.pucrs.br	10484	8225	426	16	11.634990
14	www.ufsc.br	443	341	40	10	6.890824
15	www.ufba.br	1892	1446	148	9	5.684540
16	www.unitau.br	433	332	28	6	4.567059
17	www.uemg.br	253	198	122	6	4.852410
18	www.uerj.br	146	107	23	6	3.710793
19	www.unimontes.br	1242	1165	524	5	2.844203
20	www.unesp.br	413	293	73	4	2.795951
21	www.ufla.br	664	590	55	4	3.207362
22	www.ufop.br	1593	1300	123	7	5.591087
23	www.ufscar.br	1467	1123	24	7	5.077374
24	www.unifesp.br	8840	6943	118	6	4.484946
25	www.ufv.br	4860	3762	112	9	6.560644
26	www.unirio.br	209	170	18	5	3.027061
27	www.uff.br	2142	1676	55	8	5.414277
28	www.ufrrj.br	996	787	59	9	6.887434
29	www.unesc.br	51	38	14	3	2.284093
30	www.furg.br	1874	1729	306	6	5.046123
31	www.uel.br	211	188	27	4	3.933367
32	www.uepg.br	374	262	27	9	5.733368
33	www.unioeste.br	1208	987	46	8	4.988114
34	www.ufpe.br	842	641	30	10	6.956692
35	www.furb.br	175	167	64	3	2.417960
36	www.ueg.br	283	241	57	4	2.825582
37	www.uems.br	1208	1038	53	7	5.361259
38	www.upe.br	32	23	10	5	3.056459
39	www.uneb.br	600	479	51	6	4.069363
40	www.unicap.br	48	35	21	6	4.860433
41	www.uesc.br	1161	1032	503	5	3.192340
42	www.uema.br	181	159	41	3	1.979807

	Site	n	m	d	H	$E[T, \mathbf{p}]$
43	www.uespi.br	702	670	434	4	2.158152
44	www.uesb.br	500	460	318	3	2.027795
45	www.uva.br	612	534	55	5	2.852962
46	www.ufrpe.br	733	602	39	9	5.152063
47	www.ufpb.br	117	96	20	4	3.325756
48	www.uepa.br	181	157	35	4	3.006262
49	www.unitins.br	176	164	84	3	2.139179
50	www.unir.br	613	545	171	6	4.536631
51	www.ufrr.br	130	109	47	4	1.906202
52	www.ufac.br	2852	2492	145	8	4.558061
53	www.unifap.br	305	239	102	4	2.417264
54	www.ufpa.br	787	727	260	6	4.685219
55	www.ufpi.br	1100	650	48	8	5.572896
56	www.ua.edu	104	85	18	6	2.850382
57	www.alaska.edu	488	400	134	7	4.950949
58	www.arizona.edu	5704	4821	389	7	4.103460
59	www.calstate.edu	19912	16628	420	8	5.771347
60	www.colorado.edu	18307	15221	394	6	4.830470
61	www.uconn.edu	47	35	15	3	1.972741
62	www.yale.edu	16246	14373	1250	6	4.625469
63	www.fsu.edu	21296	17308	343	9	6.248347
64	www.ufl.edu	81	60	25	4	2.705313
65	www.uga.edu	11140	9679	261	5	4.525733
66	www.uchicago.edu	794	688	126	4	3.014701
67	www.indiana.edu	1964	1738	51	4	3.125306
68	www.ku.edu	561	520	120	3	2.200642
69	www.harvard.edu	257	233	53	7	4.689610
70	www.umich.edu	49737	43310	1309	6	5.002102
71	www.umt.edu	8124	6792	405	6	4.377365
72	www.unmc.edu	9087	7340	236	7	5.244113
73	www.unlv.edu	512484	493048	182	14	11.986448
74	www.princeton.edu	5279	4545	127	4	3.714349
75	www.unm.edu	8902	7756	683	5	4.259232
76	www.cornell.edu	99	81	40	5	2.743209
77	www.columbia.edu	10530	9258	376	4	3.581786
78	www.ou.edu	14113	11415	250	9	6.372947
79	www.uoregon.edu	1267	968	46	9	6.188894
80	www.cmu.edu	1696	1503	240	4	3.159945
81	www.upenn.edu	518	452	34	3	2.336106
82	www.virginia.edu	636	588	175	3	2.082653
83	www.washington.edu	10182	9185	3154	4	3.159616
84	www.stanford.edu	1494	1330	51	3	2.836349

Tabela 5.1: As instâncias obtidas dos sites de universidades brasileiras e americanas.

lizada. Apenas a qualidade das soluções obtidas para $D < H$ é afetada por estas probabilidades.

5.2 Aspectos de implementação

5.2.1 Reduzindo o número de sub-problemas

Seja $(\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p})$ um sub-problema gerado por M-PATH. Devido ao uso da técnica de programação dinâmica, M-PATH mantêm uma tabela que contêm o valor ótimo de todos os sub-problemas gerados. A seguir, discutimos como determinado sub-problema pode ser encontrado nesta tabela. Depois mostramos como reduzir o tamanho desta tabela.

Lembre que toda árvore $\mathcal{T}$ de um sub-problema gerado pode ser obtida de alguma sub-árvore T_r de T removendo os últimos i filhos de r (seguindo alguma ordem arbitrária). Utilizamos então o último filho não removido de r para identificar $\mathcal{T}$ no conjunto de todas as sub-árvores geradas. Denotamos este último filho por f. Por exemplo, no sub-problema da Figura 3.5.(a), f é o terceiro filho de r (da esquerda para a direita). Por outro lado, f é o segundo filho de r no sub-problema da Figura 3.5.(b). Se r não tem nenhum filho, então $\mathcal{T}$ não precisa ser identificada, pois o sub-problema correspondente leva a condição de parada do algoritmo M-PATH. Além disso, seja α o valor binário do vetor $\mathbf{a}$, dado por $\sum_{i=1}^{k+1} a_i \ 2^{i-1} + b \ 2^{k+1}$. Utilizamos este valor para identificar o vetor $\mathbf{a}$ no conjunto de todos os vetores gerados. Portanto, um elemento na tabela de sub-problemas do algoritmo M-PATH pode ser identificado pelo par (α, f).

Lembre que quando M-PATH decompõe um sub-problema de acordo com o caso 2, ele verifica $2^{k'}$ valores possíveis para o vetor binário $\mathbf{c}$, onde $k' = \sum_{i=1}^{k+1} a_i$ é o número de hotlinks disponíveis no caminho $\mathbf{q}$ e na raiz r de $\mathcal{T}$. Se $\mathcal{T}$ tem m hotleaves com $m < k'$, então, com base no Lema 5.1 a seguir, M-PATH assume que os últimos $k' - m$ hotlinks disponíveis de $\mathbf{q} \rightarrow r$ não precisam ser atribuídos. Para isso, ele faz estes últimos elementos de $\mathbf{a}$ iguais a zero. Como resultado, apenas 2^m possibilidades são verificadas, e muitos elementos não precisam ser armazenados na tabela de sub-problemas.

Lema 5.1 *Seja $I = (\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p})$ uma instância do P-MBH gerada como um sub-problema da árvore de entrada T, onde $\mathcal{T}$ tem m folhas e $\mathbf{q} \rightarrow r$ tem $k' < m$ nós disponíveis. Então, existe uma atribuição de hotlinks ótima para I que atribui apenas os primeiros m hotlinks disponíveis de $\mathbf{q}$.*

Prova. Considere a instância do P-MBH apresentada na Figura 5.1.(a). Seja A uma atribuição de hotlinks obtida por M-PATH. Temos na Figura 5.1.(b) uma árvore resultante desta atribuição de hotlinks. Uma sub-árvore $T^A_{u_i}$ existirá se o hotlink de q_i é utilizado, para $i = 1, \ldots, k$. Se mais de m hotlinks foram utilizados em $\mathbf{q}$, então pelo menos uma sub-árvore $T^A_{u_i}$ não possui hotleaves. Se $T^A_{u_i}$ não possui hotleaves, podemos remover de A o hotlink (q_i, u_i) sem alterar o custo da solução. Depois que os hotlinks desnecessários são removidos, suponha que o hotlink de q_i é utilizado e o hotlinks de q_{i-1} não é. Neste caso, podemos melhorar a solução inserindo o hotlink (q_{i-1}, u_i) e removendo o hotlink (q_i, u_i).

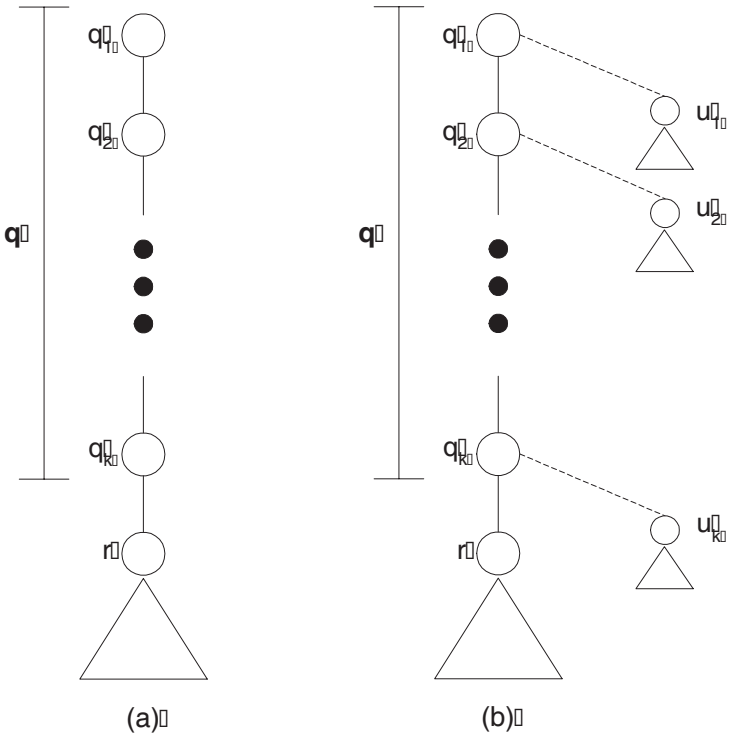

Figura 5.1: (a) Instância do P-MBH. (b) Árvore resultante da atribuição de hotlinks A.

□

Para armazenar e recuperar eficientemente os valores ótimos dos sub-problemas gerados por M-PATH, propomos um método para indexar a tabela de sub-problemas. Dada uma instância $I = (\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p})$ do P-MBH onde $\mathcal{T}$ tem m hotleaves, seja β o número de vetores $\mathbf{a}'$ com não mais que m hotlinks disponíveis e valor binário menor que $\mathbf{a}$. Se $\mathbf{a}$ também não tem mais do que m hotlinks disponíveis, então usamos o par (β, f) para identificar I na tabela de sub-problemas. Por exemplo, se $m = 2$ e $\mathbf{a} = (1, 0, 0, 0, 0)$, então $\alpha = 16$ e $\beta = 14$ pois cada um dos vetores $(0, 1, 1, 1, 0)$ e $(0, 1, 1, 1, 1)$ tem mais de dois hotlinks disponíveis e valor binário menor que 16.

A seguir, mostraremos como calcular recursivamente o valor de β como função de $\mathbf{a}$ e m. Se $\mathbf{a} = (0, \mathbf{a}')$, então $\beta(\mathbf{a}, m) = \beta(\mathbf{a}', m)$. Caso contrário, se $\mathbf{a} = (1, \mathbf{a}')$, então $\beta(\mathbf{a}, m) = \beta(\mathbf{a}', m-1) + \gamma(|\mathbf{a}|, m)$, onde $\gamma(|\mathbf{a}|, m)$ é o número total de vetores com $|\mathbf{a}|$ elementos, não mais que m hotlinks disponíveis, e $a_1 = 0$. A recursão para quando $|\mathbf{a}| = 1$. Neste caso, sempre retornaremos o valor b. Para melhorar a performance deste cálculo, obtemos o valor de $\gamma(|\mathbf{a}|, m) = 2 \sum_{i=0}^{m} \binom{|\mathbf{a}|-2}{i}$, para $|\mathbf{a}| = 1, \ldots, D$ e $m = 1, \ldots, |\mathbf{a}| - 2$, de outra tabela que é construída em uma fase de pré-processamento.

5.2.2
Enfraquecendo a restrição de altura

Sejam $(\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p})$ uma instância do P-MBH, e $z = \min\{i \mid a_i = 1\} - 1$. Observamos que $E^*(\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p}) = z\ \mathbf{p}(\mathcal{T}) + E^*(\mathbf{q}', \mathbf{a}', \mathcal{T}, \mathbf{p})$, onde $\mathbf{a}'$ ($\mathbf{q}'$) é obtido de $\mathbf{a}$ ($\mathbf{q}$) removendo-se os z primeiros elementos, e $\mathbf{p}(\mathcal{T})$ denota a soma das probabilidades de todas as hotleaves em $\mathcal{T}$. Por exemplo, se temos $\mathbf{a} = (0, 0, 0, 1, 0, 1, 1)$, então $z = 3$ e $\mathbf{a}' = (1, 0, 1, 1)$. Neste caso, como nenhum hotlink pode ser atribuído em q_1, q_2 ou q_3, estes nós apenas provocam um acréscimo constante no custo da solução ótima. Como podemos utilizar o valor de $E^*(\mathbf{q}', \mathbf{a}', \mathcal{T}, \mathbf{p})$ para calcular $E^*(\mathbf{q}, \mathbf{a}, \mathcal{T}, \mathbf{p})$, precisamos armazenar apenas $E^*(\mathbf{q}', \mathbf{a}', \mathcal{T}, \mathbf{p})$ na tabela.

A consequência imediata desta melhoria é a redução da necessidade de tempo e espaço para a execução do algoritmo. Entretanto, existe uma consequência adicional. Note que descartamos apenas sub-problemas com $|\mathbf{q}'| > D$. Como resultado, esta melhoria fornece a melhor solução possível A_D^* com a restrição de que nenhum sub-problema com $|\mathbf{q}'| > D$ é utilizado para construir $T^{A_D^*}$. Observe que esta restrição é mais fraca que $H(T^{A_D^*}) \leq D$.

5.2.3
Descarte de memória

Como a tabela da programação dinâmica pode ficar muito grande, desenvolvemos um mecanismo para descarte de memória. A idéia é utilizar uma abordagem bottom-up para encontrar o valor ótimo de cada sub-problema com raiz em u, utilizando os valores ótimos dos sub-problemas com raiz nos filhos de u. A memória que armazena os valores ótimos de um sub-problema com raiz em um filho de u é descartada quando os valores ótimos

para o filho seguinte são calculados. Quando os valores ótimos de todos os sub-problema com raiz em u são calculados, toda a memória alocada para guardar valores ótimos para filhos de u deve estar descartada.

Suponha que vamos calcular os valores ótimos de todos os sub-problemas onde $\mathcal{T}$ tem raiz em um nó u no nível l, com s filhos $(v_1, \ldots, v_s)$. Neste caso, temos um sub-problema para cada caminho com comprimento menor ou igual a l, para cada combinação de hotlinks disponíveis, e para cada valor de b (se u pode ou não receber hotlink).

Seja $(\mathbf{q}, \mathbf{a}(0), T_u, \mathbf{p})$ um destes sub-problemas considerados, com $|\mathbf{q}| = k \leq l$. Seja também $T_u^{(i)}$ a sub-árvore obtida de T_u removendo-se os $s - i$ últimos filhos de u. Note que $E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(1)}, \mathbf{p}) = E^*(\mathbf{q} \rightarrow v_1, \mathbf{a}(1,1), T_{v_1}, \mathbf{p})$. Além disso, $E^*(\mathbf{q}, \mathbf{a}, T_u, \mathbf{p}) = E^*(\mathbf{q}, \mathbf{a}, T_u^{(s)}, \mathbf{p})$.

Suponha que já calculamos $E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(i-1)}, \mathbf{p})$. Para obter o valor de $E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(i)}, \mathbf{p})$, testamos todas as maneiras de distribuir os hotlinks de $\mathbf{q} \rightarrow u$ entre $T_u^{(i-1)}$ e T_{v_i}. Seja C o conjunto de vetores binários definido por $C = \{(c_1, \ldots, c_{k+1}) \mid c_i \leq a_i \text{ para } i = 1, \ldots, k+1\}$. Cada $\mathbf{c} \in C$ corresponde a uma das possibilidades de selecionar os nós que permanecerão disponíveis para apontar para nós em T_{v_i}. Além disso, o vetor $\overline{\mathbf{c}} = \mathbf{a} - \mathbf{c}$ define quais nós de $\mathbf{q}$ permanecerão disponíveis para apontar para nós em $T_u^{(i-1)}$. Como $b = 0$, não é necessário considerar o caso 1 do algoritmo M-PATH. Assim, utilizamos apenas a equação do caso 2 abaixo.

$$E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(i)}, \mathbf{p}) = \min_{\mathbf{c} \in C} \{E^*(\mathbf{q}, \overline{\mathbf{c}}(0), T_u^{(i-1)}, \mathbf{p}) + E^*(\mathbf{q} \rightarrow v_i, \mathbf{c}(1,1), T_{v_i}, \mathbf{p})\}.$$

Todos os sub-problemas com $b = 0$ devem ser calculados antes dos sub-problemas com $b = 1$, pois os valores ótimos para os problemas com $b = 0$ são utilizado no caso 1 do algoritmo M-PATH, como pode ser verificado na equação do caso 1 abaixo.

$$E^*(\mathbf{q}, \mathbf{a}(1), T_u^{(i)}, \mathbf{p}) = \min_{j \in \{1,\ldots,k\} \mid a_j = 1} \{E^*(\mathbf{q}_j, \mathbf{a}_{j-1}(0, a_{k+1}, 0), T_u^{(i)}, \mathbf{p})\}$$

A junção dos resultados dos casos 1 e 2 permanece inalterada com o descarte de memória. Portanto, como pode ser observado nas equações acima, podemos calcular todos os sub-problemas com raiz de $\mathcal{T}$ em u utilizando as soluções dos sub-problemas com raiz de $\mathcal{T}$ em filhos de u.

Depois que calculamos os valores de $E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(i)}, \mathbf{p})$ e $E^*(\mathbf{q}, \mathbf{a}(1), T_u^{(i)}, \mathbf{p})$ para todo $\mathbf{q}$ e $\mathbf{a}$, podemos descartar a memória que

armazena os valores de $E^*(\mathbf{q}, \mathbf{a}(1,1), T_{v_i}, \mathbf{p})$. A memória utilizada para os valores de $E^*(\mathbf{q}, \mathbf{a}(0), T_u^{(i-1)}, \mathbf{p})$ e $E^*(\mathbf{q}, \mathbf{a}(1), T_u^{(i-1)}, \mathbf{p})$ também pode ser descartada.

Recuperando os hotlinks atribuídos

Com o descarte de memória, ao final da execução do algoritmo não dispomos da tabela da programação dinâmica para recuperar os hotlinks atribuídos. Portanto, utilizamos este algoritmo com descarte de memória apenas para obter o valor ótimo de um sub-problema. Chamamos este algoritmo de OPT-VAL. Chamamos de MF-PATH o algoritmo que deve recuperar os hotlinks. Denotamos por $E^*[T]$ o valor de $E^*(\mathbf{q}, \mathbf{a}, T, \mathbf{p})$ quando $|\mathbf{q}| = 0$. Dada uma árvore T, OPT-VAL fornece o valor de $E^*[T]$.

Quando a tabela da programação dinâmica excede o máximo de memória disponível, MF-PATH utiliza OPT-VAL para auxiliar na descoberta de um nó que deve receber o hotlink da raiz. Depois que este hotlink é atribuído, MF-PATH é chamado recursivamente para os sub-problemas gerados. Por outro lado, se a tabela da programação dinâmica cabe na memória disponível, MF-PATH chama o algoritmo M-PATH, pois este algoritmo é mais eficiente na recuperação dos hotlinks.

Suponha que T, com raiz em r, é a árvore de entrada para MF-PATH. Uma maneira simples de descobrir qual nó deve receber o hotlink da raiz é testar cada nó u, chamando OPT-VAL para T_u e para $T - T_u$. O nó escolhido deve minimizar $E^*[T - T_u] + (E^*[T_u] + p_u)$, onde p_u é a soma das probabilidade das folhas descendentes de u em T.

Para tornar o MF-PATH mais eficiente, devemos reduzir o número de chamadas a OPT-VAL. Primeiro, adaptamos OPT-VAL para guardar em cada nó u de T o valor de $E^*[T_u]$ (não apenas retornar $E^*[T]$). Com isso, quanto testamos um nó u, é suficiente chamar OPT-VAL apenas para $T - T_u$. Para evitar testar todos os nós para saber qual deles deve receber o hotlink, utilizamos a expressão abaixo. Sabemos que um nó u que recebe o hotlink da raiz deve satisfazer a condição

$$E^*[T] = E^*[T_i - T_u] + E^*[T_u] + \sum_{j \in S_r - \{i\}} E^*[T_j] + p_r,$$

onde i é o ancestral de u que é filho de r.

Devemos considerar também a ordem dos candidatos testados. Testamos os nós em ordem decrescente de ganho, como sugere a heurística GREEDY-BFS. O que motivou este critério foi a qualidade das soluções

obtidas por este algoritmo nos experimentos, cujos resultados são apresentado a seguir.

5.3 Resultados

Para cada uma das 84 instâncias, executamos os quatro algoritmos: GREEDY-BFS [14], KRANAKIS [12], M-PATH e a implementação do modelo em programação inteira no XPRESS [2] (MIP). Dispondo de 16MB para a tabela da programação dinâmica, M-PATH encontrou a solução ótima de 79 instâncias. Com 23MB, M-PATH é capaz de fornecer a solução ótima para 82 instâncias (apenas as instâncias 13 e 73 exigem mais memória). Para as 5 instâncias que não foram resolvidas com 16MB (instâncias 13, 63, 70, 73 e 78), aplicamos o algoritmo MF-PATH para reduzir gradualmente a memória até 1MB. Entretanto, para as instâncias 13 e 73, MF-PATH levou mais de uma hora. Portanto, para estas instâncias, utilizamos o algoritmo M-PATH variando o valor de D, limitando a memória utilizada em 1GB.

O ganho de um algoritmo Alg para uma instância $(T, \mathbf{p})$ é dada por $(E[T, \mathbf{p}] - E[T^A, \mathbf{p}])/E[T, \mathbf{p}]$, onde A é o conjunto de hotlinks que Alg adiciona na árvore T. O gráfico da Figura 5.2 mostra o ganho médio de cada algoritmo como uma função da altura da árvore de entrada. Por outro lado, o gráfico da Figura 5.3 mostra o custo da solução de cada algoritmo em relação ao custo do algoritmo M-PATH. Utilizamos nestes gráficos as 82 instâncias em que M-PATH forneceu solução ótima dispondo de 23MB. Observamos que as soluções dos algoritmos GREEDY-BFS e KRANAKIS afastam-se do ótimo com o crescimento de H. Observamos também que, como em [14], as soluções dadas por GREEDY-BFS são melhores que aquelas do KRANAKIS. Porém, damos a informação adicional de que, para nossas instâncias, as soluções do GREEDY-BFS afastam-se no máximo 5% da solução ótima (na média).

Apresentamos na Tabela 5.2 o ganho ótimo de cada instância, o ganho e o tempo de execução dos algoritmos GREEDY-BFS e KRANAKIS, e o tempo de execução do MIP. Para o M-PATH dispondo de 16MB, a tabela fornece o tempo de execução, o tamanho da tabela em KB, e a redução percentual de memória com as melhorias (redução do número de sub-problemas e enfraquecimento da restrição de altura). Observamos que em termos de performance, ambos GREEDY-BFS e KRANAKIS resolveram todas as instâncias em menos de 2 segundos. O XPRESS encontrou a solução ótima de 23 instâncias. As 63 instâncias restantes (indicadas com um "-")

não terminaram em menos de uma hora. Dentre as instâncias resolvidas, 9 levaram menos de 2 segundos. M-PATH gastou menos de 2 segundos em cada uma das 79 instâncias que puderam ser resolvidas utilizando 16MB para a tabela da programação dinâmica. Indicamos com "-"as 5 instâncias que exigem mais de 16MB de memória. Concluímos que a redução média da memória utilizada por M-PATH, devido às melhorias, foi de 31.3% com desvio padrão de 12.5%. Esta redução chegou a 68.5% na instância 55.

Na Tabela 5.3 apresentamos o tempo de execução para diferentes quantidades de memória disponível para o algoritmo MF-PATH, para as instâncias 63, 70 e 78. Na última linha, temos memória suficiente para resolver cada instância utilizando o algoritmo M-PATH. Na quarta linha, reduzimos em um byte esta quantidade de memória, impedindo que MF-PATH chame M-PATH para a árvore de entrada. Seja M a quantidade máxima de memória para resolver os sub-problemas gerados por MF-PATH na linha 4. Assim, chamamos MF-PATH com $M - 1$ bytes na linha 3, impedindo que MF-PATH chame M-PATH para os sub-problemas que consomem M bytes. Prosseguimos desta maneira até que a memória necessária fosse menos de 1MB. Note que o tempo de execução fica bastante reduzido quando a instância pode ser resolvida por M-PATH. Porém, quando não foi possível utilizar M-PATH na instância inteira, o tempo de execução aumentou lentamente com a redução da memória.

Foi necessário reduzir a qualidade da solução das instâncias 13 e 73, pois elas exigiram mais de uma hora de execução do algoritmo MF-PATH dispondo de 256MB. Para estas instâncias, apresentamos na Tabela 5.4 a memória utilizada, o tempo de execução, o ganho e a altura da árvore melhorada produzida pela execução de M-PATH, para o parâmetro D variando de 1 até 16. Limitamos a memória disponível em 1GB. Note que M-PATH encontrou a solução ótima da instância 13, pois sua altura é 16. Para auxiliar a análise, construímos os gráficos do uso de memória e do ganho de cada instância na Figura 5.4. Observamos no gráfico que o consumo de memória cresce de forma acentuada com o aumento de D (bem como o tempo de execução, como pode ser visto na tabela). Entretanto, devido à melhoria de enfraquecimento da restrição de altura, o ganho cresce rapidamente, tendendo a convergir para o ótimo com valores pequenos de D. Observamos que, sem enfraquecer a restrição de altura, M-PATH apenas encontra uma solução A quando $H(T^A) \leq D$. Após enfraquecer a restrição de altura, porém, isto não é necessariamente verdadeiro. Na Tabela 5.4, todas as soluções A com $D < 10$ tiveram $H(T^A) > D$. Embora GREEDY-BFS tenha encontrado boas soluções para todas as instâncias, com $D = 4$

M-PATH superou este algoritmo na instância 73. Na instância 13, M-PATH superou o algoritmo GREEDY-BFS com $D = 5$.

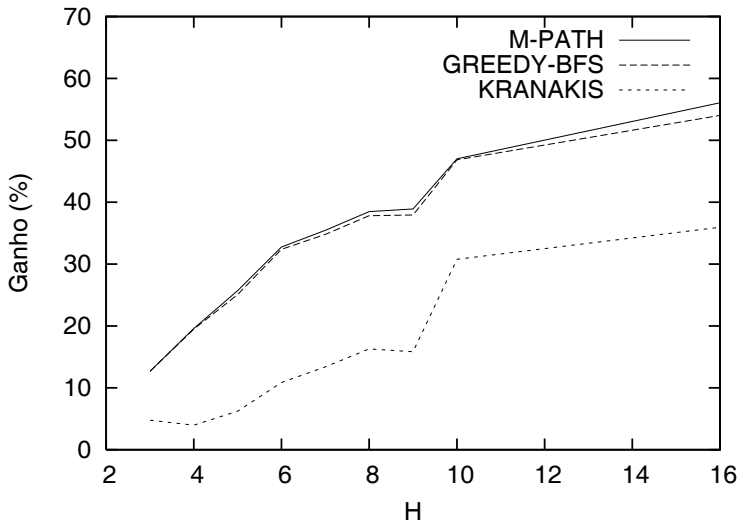

Figura 5.2: Ganho médio (%) de cada algoritmo em função de H.

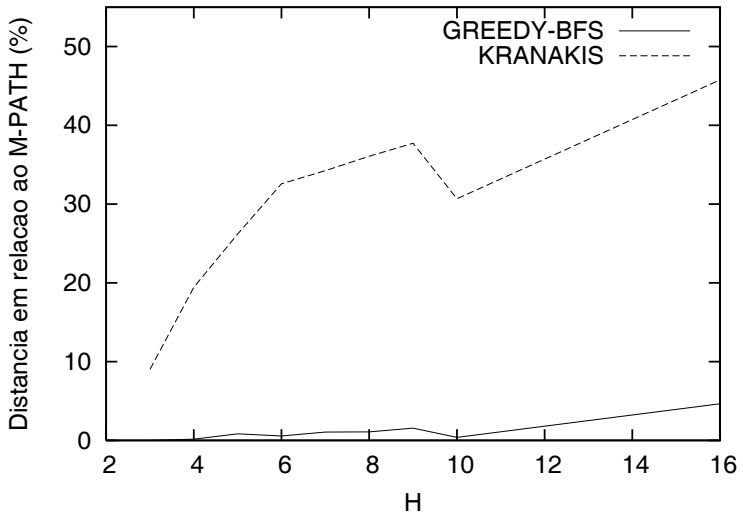

Figura 5.3: Diferença da solução de cada algoritmo em relação a solução do algoritmo M-PATH.

		GREEDY-BFS		KRANAKIS		MIP	M-PATH 16MB		
Instância	Ótimo %	%	seg	%	seg	seg	seg	KB	%
1	29.4	29.4	0.01	7.8	0.01	-	0.01	100	23.5
2	41.2	41.0	0.01	21.9	0.01	70.27	0.02	143	33.1
3	15.2	15.2	0.01	6.8	0.01	-	0.01	104	21.4
4	33.5	33.5	0.01	17.2	0.01	-	0.07	589	38.7
5	40.1	39.6	0.01	16.1	0.01	-	0.07	517	38.2
6	30.9	29.3	0.01	6.7	0.01	-	0.06	554	39.9
7	25.4	24.6	0.01	5.9	0.01	42.72	0.01	42	46.9
8	18.4	18.4	0.01	1.4	0.01	-	0.01	48	15.9
9	23.5	22.4	0.01	1.0	0.01	-	0.04	659	26.0
10	40.0	39.4	0.01	19.7	0.01	-	0.09	488	57.0
11	14.1	14.1	0.01	0.4	0.01	-	0.01	117	27.6
12	27.0	26.2	0.01	1.2	0.01	-	0.01	90	21.4
13	-	54.0	0.04	36.0	0.03	-	-	-	-
14	46.1	45.7	0.01	30.0	0.01	-	0.34	1031	46.4
15	37.9	35.8	0.01	13.0	0.01	-	0.23	1176	46.4
16	29.5	29.4	0.01	8.8	0.01	-	0.01	107	40.9
17	48.8	48.8	0.01	29.2	0.01	16.43	0.01	81	34.9
18	34.6	34.5	0.01	18.4	0.01	3.11	0.01	24	40.7
19	24.8	24.8	0.01	22.8	0.01	-	0.01	146	10.6
20	19.3	19.3	0.01	2.3	0.01	-	0.01	33	24.3
21	23.7	23.6	0.01	1.7	0.01	-	0.01	70	35.2
22	35.6	35.0	0.01	13.5	0.01	-	0.07	771	33.3
23	37.5	37.3	0.01	14.0	0.01	-	0.06	463	44.5
24	25.7	25.1	0.01	4.6	0.03	-	0.23	2141	40.3
25	36.2	36.1	0.02	9.6	0.01	-	1.29	5384	51.0
26	20.4	19.4	0.01	7.6	0.01	8.36	0.01	28	25.3
27	35.4	33.6	0.01	7.6	0.01	-	0.16	1046	45.9
28	44.5	43.0	0.01	16.1	0.01	-	0.28	1216	42.6
29	23.7	23.7	0.01	21.0	0.01	0.14	0.01	4	22.4
30	41.3	41.3	0.01	19.4	0.01	-	0.06	835	16.6
31	28.3	28.3	0.01	3.3	0.01	9.21	0.01	30	39.2
32	46.7	45.3	0.01	26.1	0.01	97.30	0.02	146	65.2
33	39.7	39.0	0.01	18.7	0.01	-	0.06	549	27.4
34	48.0	47.9	0.01	31.5	0.01	-	0.95	1869	49.6
35	12.1	12.1	0.01	4.8	0.01	5.42	0.01	12	32.5
36	13.5	12.6	0.01	4.3	0.01	20.30	0.01	29	19.5
37	36.8	35.1	0.01	18.5	0.01	-	0.06	623	20.2
38	33.9	33.9	0.01	8.8	0.01	0.04	0.01	3	33.3
39	27.6	26.5	0.01	3.4	0.01	-	0.01	120	40.2
40	51.9	51.9	0.01	29.8	0.01	0.21	0.01	14	31.1
41	24.9	24.9	0.01	1.1	0.01	-	0.01	166	25.7
42	9.5	9.5	0.01	3.0	0.01	5.32	0.01	11	13.2

		GREEDY-BFS		KRANAKIS		MIP	M-PATH 16MB		
Instância	Ótimo %	%	seg	%	seg	seg	seg	KB	%
43	9.9	9.9	0.01	6.5	0.01	-	0.01	52	15.3
44	8.7	8.7	0.01	7.3	0.01	-	0.01	30	9.1
45	20.9	20.4	0.01	5.2	0.01	-	0.01	64	15.5
46	29.9	29.9	0.01	4.5	0.01	-	0.09	469	55.6
47	23.2	23.2	0.01	8.8	0.01	1.65	0.01	14	36.7
48	17.8	17.8	0.01	3.9	0.01	5.52	0.01	21	23.0
49	11.2	11.2	0.01	1.2	0.01	5.23	0.01	12	19.5
50	32.0	32.0	0.01	8.6	0.01	-	0.01	197	15.9
51	11.4	11.4	0.01	5.0	0.01	1.95	0.01	8	17.1
52	36.3	36.3	0.01	14.3	0.01	-	0.08	919	25.5
53	20.1	20.1	0.01	3.5	0.01	24.23	0.01	20	23.6
54	37.7	37.7	0.01	3.7	0.01	-	0.02	234	32.2
55	46.7	46.6	0.01	24.9	0.01	-	0.04	370	68.5
56	27.1	27.1	0.01	20.7	0.01	1.20	0.01	16	22.2
57	40.0	39.5	0.01	14.1	0.01	-	0.02	199	33.6
58	25.4	24.9	0.02	6.0	0.02	-	0.08	1164	30.7
59	34.1	33.5	0.07	15.6	0.06	-	1.64	12858	35.6
60	28.0	27.9	0.06	0.8	0.05	-	0.53	5587	31.0
61	12.4	12.4	0.01	4.0	0.01	0.11	0.01	3	25.0
62	24.7	24.4	0.04	4.2	0.05	-	0.43	5386	19.6
63	35.2	33.9	0.06	15.6	0.05	-	-	-	-
64	27.1	27.1	0.01	22.3	0.01	0.51	0.01	6	23.8
65	27.6	26.5	0.03	4.5	0.02	-	0.26	2583	38.9
66	18.9	18.9	0.01	2.7	0.01	-	0.01	84	23.3
67	18.7	18.7	0.01	0.9	0.01	-	0.01	218	19.7
68	9.2	9.2	0.01	1.5	0.01	-	0.01	35	24.3
69	35.4	35.2	0.01	18.8	0.01	18.88	0.01	113	14.3
70	26.2	24.4	0.15	3.1	0.14	-	-	-	-
71	29.7	28.8	0.02	5.0	0.03	-	0.18	1902	34.6
72	30.3	29.8	0.03	3.7	0.03	-	0.46	3548	38.5
73	-	46.4	1.77	23.6	1.27	-	-	-	-
74	18.7	18.0	0.01	0.7	0.02	-	0.07	679	39.2
75	24.3	23.3	0.02	2.3	0.02	-	0.20	1964	37.2
76	27.1	27.1	0.01	7.6	0.01	1.00	0.01	9	27.4
77	20.8	20.8	0.04	0.3	0.02	-	0.13	1256	33.2
78	36.0	35.5	0.04	15.4	0.03	-	-	-	-
79	42.4	41.3	0.01	21.9	0.01	-	0.30	1195	52.6
80	16.2	16.1	0.01	1.0	0.01	-	0.01	196	26.7
81	8.9	8.9	0.01	6.4	0.01	-	0.01	35	29.8
82	7.6	7.6	0.01	1.4	0.01	-	0.01	39	12.6
83	22.7	22.7	0.03	0.1	0.03	-	0.11	1165	19.6
84	18.6	18.6	0.01	1.5	0.01	-	0.02	101	35.3

Tabela 5.2: Ganho ótimo de cada instância, ganho e tempo de execução dos algoritmos GREEDY-BFS e KRANAKIS, tempo de execução do MIP. Para o M-PATH dispondo de 16MB para armazenar a tabela da programação dinâmica, o tempo de execução, o tamanho da tabela em KB, e a redução percentual de memória com as melhorias (redução do número de subproblemas e enfraquecimento da restrição de altura).

63		70		78	
Bytes	seg	Bytes	seg	Bytes	seg
1017375	67.03	980239	12.56	910655	27.67
1298895	67.14	1210663	12.48	1497887	27.12
10929159	66.76	4843967	12.28	2785615	26.63
23812983	60.34	18585991	11.66	17434399	23.79
23812984	6.32	18585992	2.22	17434400	4.23

Tabela 5.3: Tempo de execução do algoritmo MF-PATH e memória disponível para a tabela da programação dinâmica, para as instâncias 63, 70 e 78.

	13				73			
D	Bytes	seg	%	$H(T^A)$	Bytes	seg	%	$H(T^A)$
1	202496	0.06	0.0	16	17221184	3.41	5.4	14
2	389720	0.07	5.4	16	34403032	5.57	35.1	13
3	737000	0.09	40.0	12	68601152	21.08	44.4	13
4	1388144	0.14	53.5	10	136545248	215.93	47.8	13
5	2613112	0.33	56.0	11	271489072	3710.06	48.6	14
6	4931712	0.88	56.1	10	539586808	>1h	-	-
7	9273088	2.37	56.1	10	1072312568	>1h	-	-
8	17338376	6.47	56.1	10	-	-	-	-
9	31712192	16.93	56.1	10	-	-	-	-
10	56348096	44.27	56.1	10	-	-	-	-
11	96861560	116.93	56.1	10	-	-	-	-
12	161707600	253.29	56.1	10	-	-	-	-
13	240977872	508.63	56.1	10	-	-	-	-
14	337388320	1011.56	56.1	10	-	-	-	-
15	453558280	1661.48	56.1	10	-	-	-	-
16	487713224	1654.08	56.1	10	-	-	-	-

Tabela 5.4: Memória utilizada na tabela da programação dinâmica, tempo de execução, ganho e a altura da árvore melhorada resultante do algoritmo M-PATH, para D variando de 1 até 16, para as instâncias 13 e 73. Memória disponível limitada em 1GB. Tempo de execução limitado em aproximadamente 1 hora. O "-"indica que o valor não está disponível, pois algum limite foi violado.

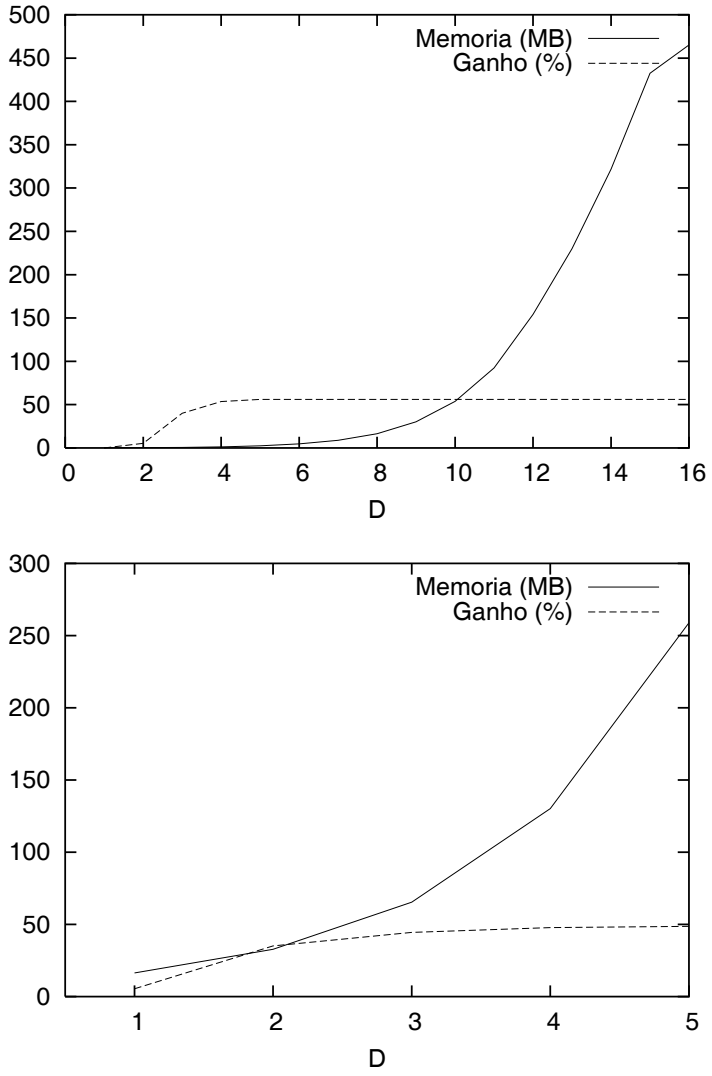

Figura 5.4: Ganho (%) e uso de memória (MB) do algoritmo M-PATH, variando o parâmetro D, para as instâncias 13 (em cima) e 73 (em baixo).