4
Algoritmos Implementados

Nesta seção descrevemos alguns algoritmos conhecidos e outros novos, que foram usados na avaliação experimental descrita no Capítulo 5.

Os algoritmos que implementamos são:

(i) algoritmo Guloso: discutido em (MSVV05) e (GM08);

(ii) algoritmo AdWords: proposto em (MSVV05);

(iii) algoritmo Primal-Dual: proposto em (BJN07);

(iv) algoritmo Preditivo: proposto em nossa dissertação;

(v) algoritmo Aleatório: proposto em nossa dissertação.

Os três primeiros foram adaptados dos artigos citados para utilizar, em lugar da restrição do orçamento, a restrição do limite de exibições por anunciante, conforme especificamos na Seção 3.1. Os dois últimos, propostos em nossa dissertação, consideram que a frequência das consultas obedece a uma certa regularidade na maior parte dos dias e utilizam estas informações como base para a predição.

4.1
Algoritmo Guloso

Neste algoritmo, para cada nova consulta, são exibidos os anunciantes com as maiores ofertas para aquela consulta. Em nossa implementação, descrita na Figura 4.1, fizemos duas adaptações para solucionar os casos de empate.

Este é um algoritmo de fácil implementação e muito rápido no tempo de execução. Quando nenhum anunciante atinge seu limite máximo de exibições, este algoritmo sempre encontra a solução ótima. Em diversos casos, quando poucos anunciantes atingem seu limite, também tem um desempenho muito.

Por outro lado, é extremamente dependente da ordem de chegada das consultas. Uma análise de pior caso demonstra que sua competitividade é 1/2, similar à encontrada por (MSVV05) considerando os limites de orçamento.

Ilustraremos com o exemplo a seguir estas duas afirmativas.

Algoritmo 1 Algoritmo Guloso

1: Receba a consulta j
2: Exiba os T anunciantes com as maiores ofertas para a consulta j, que não tenham atingido seu limite de exibições.
3: No caso de empate no valor da oferta, escolha o anunciante com maior disponibilidade de exibições.
4: Em caso de novo empate, escolha aleatoriamente.

Figura 4.1: Algoritmo Guloso para seleção de links

EXEMPLO: considere que temos apenas dois anunciantes (a_1 e a_2), dois tipos de consulta (q_1 e q_2) e que exibiremos no máximo um anunciante por consulta ($T = 1$). As ofertas dos anunciantes e o limite de exibições são descritos na Tabela 4.1.

	q_1	q_2	Limites
a_1	10	10	1000
a_2	9	0	1000

Tabela 4.1: Ofertas e limites do exemplo de algoritmo guloso

Suponha um dia com 1000 consultas q_1 e 1000 consultas q_2. Nas condições descritas, independente da ordem de chegada, a solução ótima será 1900. Veremos em seguida o desempenho do algoritmo guloso em dois casos especiais.

(i) Chegam 1000 consultas q_2 e depois 1000 consultas q_1:

Atribuição	Receita
1000 consultas q_2 para o anunciante a_1	1000
1000 consultas q_1 para o anunciante a_2	900
Receita total	1900

Tabela 4.2: Exemplo de melhor caso do algoritmo Guloso

(ii) Chegam 1000 consultas q_1 e depois 1000 consultas q_2:

Atribuição	Receita
1000 consultas q_1 para o anunciante a_1	1000
Nenhuma consulta q_2 atribuída aos anunciantes	0
Receita total	1000

Tabela 4.3: Exemplo de pior caso do algoritmo Guloso

No caso (i), o algoritmo atinge a solução ótima e ambos os anunciantes atingem seus limites de exibição conforme atribuição na Tabela 4.2. Já no caso (ii), o resultado é pouco mais que 1/2 da solução ideal, como vemos na Tabela 4.3, embora apenas o anunciante a_1 tenha atingido o limite de exibições.

Pelo exemplo anterior fica claro que o desempenho deste algoritmo está ligado diretamente a ordem de chegada das consultas. Utilizando este argumento e fazendo a análise da média de todas as permutações possíveis, (GM08) prova que a competitividade deste algoritmo é $1 - 1/e$. Considerando que as consultas chegam sempre de forma aleatória, esta análise pode ser considerada mais realista que a análise de pior caso.

4.2 Algoritmo AdWords

Este algoritmo, apresentado em (MSVV05), mescla as idéias do algoritmo Guloso e do algoritmo BALANCE, apresentado em (KP00) para o problema de *b-matching*. O primeiro favorece os anunciantes com as maiores ofertas, enquanto o segundo os anunciantes com maiores orçamentos disponíveis a cada iteração. A principal idéia por trás deste algoritmo é evitar a armadilha do algoritmo Guloso de esgotar rapidamente o orçamento de um anunciante que tem interesse em vários tipos de consultas. Quanto mais anunciantes tiverem orçamento até o fim do dia, maior a chance de uma consulta ser atribuída a algum anunciante, mas sem perder de vista o aproveitamento das melhores ofertas.

Baseado nesta idéia, os autores buscaram uma maneira de balancear o uso destes dois fatores. A cada nova consulta o algoritmo seleciona o anunciante que maximiza o produto da oferta b_{ij} por uma função $\psi(x)$ que leva em conta o orçamento que resta a cada anunciante i interessado na consulta.

Esta função ψ foi encontrada através do método *trade-off-revealing*, desenvolvido pelos autores, e é definida por: $\psi(x) = 1 - e^{-(1-x)}$. A cada consulta, o parâmetro x utilizado na função é: $t_i = m_i/B_i$, onde m_i é a quantia gasta até o momento pelo anunciante i e B_i é o orçamento total deste anunciante.

Em nossa implementação, descrita na Figura 4.2, adaptamos o algoritmo para utilizar em lugar da fração gasta do orçamento m_i, a fração do limite de exibições utilizado l_i, e substituímos o orçamento B_i pelo número máximo de exibições por anunciante L_i.

Algoritmo 2 Algoritmo AdWords

1: Receba a consulta j
2: Para cada anunciante i com oferta para a consulta j e que não tenha atingido o limite de exibições, calcule: $t_i = l_i/L_i$, onde L_i é o limite de exibições e l_i o número de exibições do anunciante até o momento
3: Atribua a consulta aos T anunciantes com maior produto: $b_{ij} \times \psi(t_i)$, onde $\psi(t_i) = 1 - e^{-(1-t_i)}$
4: Em caso de empate atribua ao anunciante com a maior oferta b_{ij} para aquela consulta
5: Em caso de novo empate, atribua ao anunciante com maior disponibilidade de exibições $L_i - l_i$
6: Se persistir o empate, escolha aleatoriamente.

Figura 4.2: Algoritmo AdWords para seleção de links

A seguir, apresentamos um exemplo similar ao usado para o algoritmo Guloso, que nos permite ver como varia a função ψ e como funciona o algoritmo na prática.

EXEMPLO: considere que temos apenas dois anunciantes (a_1 e a_2), dois tipos de consulta (q_1 e q_2) e que exibiremos no máximo um anunciante por consulta ($T = 1$). As ofertas dos anunciantes são as mesmas descritas na Tabela 4.1. Usaremos os limites de exibições L_1 e L_2 igual a 10 para mostrarmos todas as iterações de um dia completo.

Na Tabela 4.4 apresentamos o desempenho do algoritmo para um dia com 20 consultas, onde chegam primeiro 10 consultas q_1 e em seguida 10 consultas q_2. Na terceira coluna observamos como decresce a função ψ conforme um anunciante é exibido e na quarta coluna como é determinado o comportamento do algoritmo pela combinação da oferta com a disponibilidade. É interessante notar como já na terceira consulta um anunciante com uma oferta menor é privilegiado em função do orçamento disponível.

Na Tabela 4.5, apresentamos o desempenho relativo dos algoritmos no exemplo anterior.

Consultas		Ofertas		$\psi(l_i/L_i)$		$b_{ij} \times \psi$		Atribuições		Receita
j	q_j	a_1	a_2	a_1	a_2	a_1	a_2	a_1	a_2	acumulada
1	1	10	9	0,6321	0,6321	6,3212	5,6891	1	0	10
2	1	10	9	0,5934	0,6321	5,9343	5,6891	1	0	20
3	1	10	9	0,5507	0,6321	5,5067	5,6891	0	1	29
4	1	10	9	0,5507	0,5934	5,5067	5,3409	1	0	39
5	1	10	9	0,5034	0,5934	5,0341	5,3409	0	1	48
6	1	10	9	0,5034	0,5507	5,0341	4,9560	1	0	58
7	1	10	9	0,4512	0,5507	4,5119	4,9560	0	1	67
8	1	10	9	0,4512	0,5034	4,5119	4,5307	0	1	76
9	1	10	9	0,4512	0,4512	4,5119	4,0607	1	0	86
10	1	10	9	0,3935	0,4512	3,9347	4,0607	0	1	95
11	2	10	0	0,3935	0,3935	3,9347	0,0000	1	0	105
12	2	10	0	0,3297	0,3935	3,2968	0,0000	1	0	115
13	2	10	0	0,2592	0,3935	2,5918	0,0000	1	0	125
14	2	10	0	0,1813	0,3935	1,8127	0,0000	1	0	135
15	2	10	0	0,0952	0,3935	0,9516	0,0000	1	0	145
16	2	10	0	0,0000	0,3935	0,0000	0,0000	0	0	145
17	2	10	0	0,0000	0,3935	0,0000	0,0000	0	0	145
18	2	10	0	0,0000	0,3935	0,0000	0,0000	0	0	145
19	2	10	0	0,0000	0,3935	0,0000	0,0000	0	0	145
20	2	10	0	0,0000	0,3935	0,0000	0,0000	0	0	145
Total								10	5	145

Tabela 4.4: Exemplo de funcionamento do algoritmo AdWords

Algoritmos	Desempenho	Receita
Solução Ótima	100%	190
AdWords	76%	145
Guloso	53%	100

Tabela 4.5: Exemplo de competitividade do algoritmo AdWords

4.3 Algoritmo Primal-Dual

O método primal-dual, em algoritmos aproximativos, foi utilizado inicialmente em (GW96). Posteriormente, em (BN05), este método foi usado para resolver problemas de cobertura (primal) e empacotamento (dual) online.

Em (BJN07), os autores propõem a utilização do método primal-dual para a seleção de links patrocinados e provam a mesma competitividade esperada $1 - 1/e$ já encontrada em(MSVV05), de uma forma mais simples e direta. No artigo, o problema analisado é similar ao que descrevemos na

Seção 3.1, exceto pelo uso da restrição do orçamento máximo por anunciante, em lugar do limite de exibições e atribuindo apenas um anunciante a cada consulta ($T = 1$).

Para utilizar o método, definimos um problema de programação linear como se pudéssemos fracionar a distribuição dos anunciantes por consulta, como apresentamos no problema dual na Figura 4.3. O objetivo é maximizar o total da receita obtida a cada consulta. A primeira restrição garante que nenhuma consulta receberá mais que um anunciante e a segunda que nenhum limite de exibições dos anunciantes será ultrapassado.

Descrevemos, na Figura 4.4, um algoritmo que encontrará uma solução garantidamente com atribuições inteiras, ou seja, um ou nenhum anunciante será atribuído a cada consulta.

Dual		Primal	
Maximizar	$\sum_{j=1}^{m}\sum_{i=1}^{n} b_{ij}\ y_{ij}$	Minimizar	$\sum_{i=1}^{n} L_i x_i + \sum_{j=1}^{m} z_j$
Sujeito à:		Sujeito à:	
$\forall j = 1..m:$	$\sum_{i=1}^{n} y_{ij} \leq 1$	$\forall i, j:$	$x_i + z_j \geq b_{ij}$
$\forall i = 1..n:$	$\sum_{j=1}^{m} y_{ij} \leq L_i$	$\forall i:$	$x_i \geq 0$
$\forall i, j:$	$y_{ij} \geq 0$	$\forall j:$	$z_j \geq 0$

Figura 4.3: Solução por programação linear pelo método Primal-Dual

Conforme demonstrado em (Chv83), pelo teorema da dualidade fraca, qualquer solução primal viável é um limite superior para a solução ótima do problema dual. Logo, em nosso modelo, para todos os valores y_{ij}, x_i e z_j, que representem soluções viáveis para os problemas dual e primal, temos:

$$\sum_{j=1}^{m}\sum_{i=1}^{n} b_{ij}\ y_{ij} \leq \sum_{i=1}^{n} L_i x_i + \sum_{j=1}^{m} z_j$$

E o teorema da dualidade forte, garante que a solução ótima é aquela em que as soluções primal e dual são idênticas, ou seja, se y^*_{ij}, x^*_i e z^*_j são soluções

ótimas temos que:

$$\sum_{j=1}^{m}\sum_{i=1}^{n} b_{ij}\ y_{ij}^{*} = \sum_{i=1}^{n} L_i x_i^{*} + \sum_{j=1}^{m} z_j^{*}$$

Assim, a solução ótima está sempre no intervalo delimitado por uma solução primal e outra dual viáveis.

Baseado na teoria da dualidade, o objetivo do algoritmo descrito na Figura 4.4 é encontrar simultaneamente soluções primal e dual viáveis para a programação linear da Figura 4.3.

A intuição por trás do algoritmo é que se não deixarmos que as duas se afastem muito, estaremos sempre perto da solução ótima. A distância entre estas duas soluções e por conseguinte da solução ótima, determina a competitividade do algoritmo.

Apesar de não podermos provar a competitividade do algoritmo para o problema com limite de exibições, como veremos mais adiante neste capítulo, manter as soluções primal e dual próximas, pode nos levar na maioria dos casos a estar próximos da solução ótima. Assim, além de buscar ambas as soluções, queremos garantir que a cada iteração do algoritmo a variação do custo primal seja no máximo $1 + 1/(c-1)$ vezes a variação do lucro dual (BJN07).

Para isso, considerando o conjunto de anunciantes A com limites de exibições L_i para cada anunciante i, definimos a constante c como:

$$c = (1 + R_{max})^{1/R_{max}}, \text{ onde } R_{max} = max_{i \in A}\{1/L_i\}$$

Pela definição, podemos ver que devemos selecionar o menor limite de exibições L_i entre todos os anunciantes para encontrar o maior R_{max}. Se chamarmos este menor limite de L_{min}, podemos reescrever a constante c como:

$$c = (1 + 1/L_{min})^{L_{min}}$$

O algoritmo da Figura 4.4 foi adaptado de (BJN07), substituindo o limite de orçamento pelo de exibições por anunciante. Para ficar mais claro o mecanismo primal-dual, apresentamos o algoritmo para $T = 1$. Para $T > 1$, basta repetir a rotina *ProcessaConsulta* desconsiderando o anunciante que já foi utilizado na consulta.

Algoritmo 3 Algoritmo Primal-Dual

Receita $\leftarrow 0$
Para todo anunciante $i = 1$ até n **faça**
    $x_i \leftarrow 0$
Fim Para
Repita
    Receba consulta j
    ProcessaConsulta(j)
Até Terminar o dia

{Rotina que processa cada nova consulta}
ProcessaConsulta(j)
Atribua consulta j ao anunciante i que maximiza $b_{ij} \times (1 - x_i)$
Se $x_i < 1$ **então**
    Exiba anunciante i
    $Receita \leftarrow Receita + b_{ij}$
    $y_{ij} \leftarrow 1$
    $z_j \leftarrow b_{ij} - x_i$
    $x_i \leftarrow x_i(1 + 1/L_i) + 1/((c - 1)\ L_i)$ {Determinaremos c mais adiante}
Fim Se

Figura 4.4: Algoritmo Primal-Dual para seleção de links

Na Figura 4.4, na primeira linha da função *ProcessaConsulta*, o algoritmo atribui a consulta j ao anunciante i que maximiza $b_{ij} \times (1 - x_i)$. Como x_i cresce a cada vez que um anunciante i recebe uma consulta, significa que depreciamos o valor da oferta de um anunciante conforme ele vai sendo exibido. Este processo evita concentrarmos as consultas em um ou poucos anunciantes e esgotarmos rapidamente seu limite.

Para ilustrar o funcionamento deste algoritmo e o comportamento das variáveis dos problemas primal e dual a cada iteração do algoritmo, apresentamos na Tabela 4.6 um exemplo passo-a-passo para um problema idêntico ao usado para analisar o algoritmo AdWords.

Exemplo: considere que temos apenas dois anunciantes (a_1 e a_2), dois tipos de consulta (q_1 e q_2) e que exibiremos no máximo um anunciante por consulta ($T = 1$). As ofertas dos anunciantes são as mesmas descritas na Tabela 4.1. Usaremos os limites de exibições L_1 e L_2 iguais a 10 para mostrarmos todas as iterações de um dia completo.

Consultas		Ofertas		x_i		$b_{ij} \times (1 - x_i)$		Exibe	Soluções	
j	q_j	a_1	a_2	a_1	a_2	a_1	a_2		Primal	Dual
1	1	10	9	0,0628	0,0000	10,00	9,00	a_1	10,0	10
2	1	10	9	0,1318	0,0000	9,37	9,00	a_1	19,9	20
3	1	10	9	0,1318	0,0314	8,68	9,00	a_2	28,9	29
4	1	10	9	0,2077	0,0314	8,68	8,72	a_1	38,8	39
5	1	10	9	0,2077	0,0643	7,92	8,72	a_2	47,8	48
6	1	10	9	0,2077	0,0989	7,92	8,42	a_2	56,7	57
7	1	10	9	0,2077	0,1352	7,92	8,11	a_2	65,6	66
8	1	10	9	0,2912	0,1352	7,92	7,78	a_1	75,4	76
9	1	10	9	0,3831	0,1352	7,09	7,78	a_1	85,1	86
10	1	10	9	0,3831	0,1734	6,17	7,78	a_2	94,0	95
11	2	10	0	0,4841	0,1734	6,17	0,00	a_1	103,6	105
12	2	10	0	0,5953	0,1734	5,16	0,00	a_1	113,1	115
13	2	10	0	0,7175	0,1734	4,05	0,00	a_1	122,5	125
14	2	10	0	0,8520	0,1734	2,83	0,00	a_1	131,8	135
15	2	10	0	1,0000	0,1734	1,48	0,00	a_1	140,9	145
16	2	10	0	1,0000	0,1734	0,00	0,00		140,9	145
17	2	10	0	1,0000	0,1734	0,00	0,00		140,9	145
18	2	10	0	1,0000	0,1734	0,00	0,00		140,9	145
19	2	10	0	1,0000	0,1734	0,00	0,00		140,9	145
20	2	10	0	1,0000	0,1734	0,00	0,00		140,9	145
Total									140,9	145

Tabela 4.6: Exemplo de funcionamento do algoritmo Primal-Dual

Na Tabela 4.6, apresentamos o desempenho do algoritmo para um dia com 20 consultas, onde chegam primeiro 10 consultas q_1 e em seguida 10 consultas q_2.

Nas últimas colunas desta tabela observamos como as funções primal e dual caminham quase juntas, garantindo que o algoritmo está sempre perto da solução ótima após cada iteração. Esta solução que aqui chamamos de ótima não é a solução ótima final do problema offline, pois para isso teríamos que mudar o passado, mas da programação linear cada iteração.

Na mesma tabela, na coluna $b_{ij} \times (1 - x_i)$, estão os valores que o algoritmo considera para cada oferta na chegada de uma nova consulta. Podemos observar que a cada iteração, conforme cresce o número de exibições de um anunciante, suas ofertas são cada vez mais depreciadas na escolha do algoritmo pela melhor oferta.

Observamos que, apesar da receeita final ser a mesma obtida pelo algoritmo AdWords para este exemplo, os anunciantes atribuídos a cada consulta não são sempre os mesmos, como podemos ver na atribuição da consulta 6.

4.4 Algoritmos Preditivos

Os algoritmos que apresentamos nesta seção utilizam informações estocásticas sobre a quantidade de cada tipo de consulta para prever o que ocorrerá durante um novo dia. A motivação para desenvolver um algoritmo que utiliza este tipo de predição é a expectativa de que a frequência de consultas seja parecida em dias subsequentes e que esta informação sirva de base para a escolha do anunciante mais indicado para cada consulta.

Para utilização do algoritmo não é necessário que as condições sobre as quais este será executado sejam as mesmas dos dias anteriores. Uma variação no número de anunciantes, nos limites de exibição ou no valor das ofertas, não interfere no desempenho do algoritmo, pois a solução ótima calculada que servirá de base para a predição utiliza as informações referentes ao início do novo dia.

A Tabela 4.7 apresenta uma possível atribuição de consultas a anunciantes. Uma tabela similar a esta é utilizada pelos algoritmos propostos e é construída da seguinte forma. Primeiro, para uma faixa selecionada de dias, calculamos a média de ocorrência de consultas de cada tipo, que será a expectativa de ocorrência da consulta para o dia que se inicia. A consulta q_2, por exemplo, teve uma média de 60 consultas diárias no período. Em seguida, calculamos a solução ótima para atribuir estas consultas aos anunciantes levando em consideração as ofertas e limites no momento do cálculo. Na Tabela 4.7 vemos, por exemplo, que temos a expectativa de receber 100 consultas q_1 e, para maximizarmos a receita, 40 delas devem ser atribuídas ao anunciante a_1, 30 ao anunciante a_2 e assim por diante.

	a_1	a_2	a_3	a_4	Total
q_1	40	30	20	10	100
q_2	20	30	10	0	60
q_3	10	0	0	10	20
Total	70	60	30	20	180

Tabela 4.7: Solução ótima para utilização nos algorimos preditivos

Em nosa dissertação, propomos dois algoritmos distintos para implementar o uso de predição na atribuição das consultas aos anunciantes. O primeiro, determinístico, procura atribuir as consultas aos anunciantes imitando a distribuição ótima calculada, mas cuidando para não sobrecarregar um anunciante e deixar outro sem atribuição. O segundo algoritmo é aleatório e a solução ótima calculada serve de base para o sorteio da atribuição, privilegiando os anunciantes que tiveram o maior número de atribuições no cálculo da solução ótima nos dias anteriores.

A seguir detalhamos como funcionam ambos os algoritmos.

4.4.1 Algoritmo Preditivo

Considere que vamos iniciar um novo dia e antes de começar a receber consultas montamos uma tabela similar a Tabela 4.7. Nesta tabela, chamamos de E_j, a expectativa de ocorrência da consulta j para o dia que se inicia, e e_{ij}, o número esperado de atribuições da consulta j ao anunciante i, que encontramos calculando a solução ótima para a média de ocorrência de cada tipo de consulta nos dias anteriores.

A estratégia deste algoritmo é atribuir as consultas que chegam aos anunciantes tentando repetir a distribuição ótima prevista. Assim, ao chegar uma consulta j ela é atribuída ao anunciante com a maior razão e_{ij}/E_j. A escolha, desta forma, privilegia os anunciantes com maior ocorrência na solução ótima prevista. Esta razão vai diminuindo ao longo do dia, conforme os anunciantes vão recebendo consultas, pois a cada iteração decrementamos e_{ij}, diminuindo da tabela de expectativas uma previsão que já ocorreu. Este algoritmo está descrito na Figura 4.5.

Do modo que montamos o algoritmo, eventualmente, uma consulta pode não ser atendida, mesmo que exista, no momento da sua chegada, um anunciante disponível para ela. Esta é uma das caractísticas da solução ótima: deixar de atender algumas consultas ao longo do dia para atender a outras de maior receita mais adiante. Este comportamento o diferencia fortemente dos algoritmos apresentados anteriormente que, mesmo com estratégias diferentes, sempre atendem uma consulta se existe um anunciante disponível. Consideramos disponível um anunciante que fez uma oferta por esta consulta e que não atingiu o limite de exibições.

Algoritmo 4 Algoritmo Preditivo (determinístico)

Repita

Receba consulta j

Atribua consulta j aos T anunciantes que maximizam e_{ij}/E_j

Para todo anunciante i que recebeu a consulta j **faça**

$e_{ij} \leftarrow e_{ij} - 1$

Fim Para

Até Terminar o dia

Figura 4.5: Algoritmo Preditivo para seleção de links

4.4.2 Algoritmo Preditivo Aleatório

Como no algoritmo anterior, considere que montamos uma tabela com as expectativas sobre as consultas antes de iniciar o dia. A estratégia agora não é determinar o anunciante a ser selecionado pela razão e_{ij}/E_j, mas dar mais chance a este anunciante de ser sorteado a cada rodada.

Diferentemente do algoritmo Preditivo, sempre que possível, este algoritmo irá atribuir a consulta a um anunciante, porque quando é feita a atribuição, a tabela de predição não tem a expectativa de cada consulta modificada. Só são deixadas de lado aquelas consultas que, na solução ótima, não foram atribuídas a nenhum anunciante ou que não tenham nenhum anunciante disponível para elas.

O comportamento aleatório do algoritmo determina que este não tente imitar a solução ótima, mas utilize esta solução como base para direcionar os sorteios dos anunciantes, como vemos no algoritmo descrito na Figura 4.6.

Algoritmo 5 Algoritmo Aleatório (com uso de predição)

Repita

Receba consulta j

Sorteie T anunciantes definindo uma probabilidade e_{ij}/E_j para cada anunciante i

Atribua a consulta j aos T anunciantes sorteados

Até Terminar o dia

Figura 4.6: Algoritmo Aleatório para seleção de links