4
Política de Tempo Igualmente Espaçada por Página

Este capítulo inicia definindo a política de tempo igualmente espaçada por página, e demonstrando que esta política é ótima em termos do *freshness* do repositório quando não precisamos respeitar a restrição de *politeness*. Esta política de tempo também é ótima para outras métricas, conforme demonstrado na literatura (Wol02, Cof98). Entretanto, podemos facilmente construir um exemplo onde esta política provoca violação da restrição de *politeness*.

Devido a esta otimalidade para o caso em que não temos a restrição de *politeness*, vale a pena investigar políticas que realizam ajustes sobre o escalonamento igualmente espaçado por páginas de modo a respeitar a restrição de *politeness*. A política DELAYED, definida na Seção 4.2, insere em cada revisitação que viola a restrição de *politeness* o menor atraso necessário para que esta restrição seja respeitada. A Seção 4.2 fornece também um algoritmo eficiente para a política DELAYED. Embora nenhum fator de aproximação para o *freshness* do repositório seja demonstrado para esta política, os experimentos apresentados neste capítulo indicam que esta política fornece um *freshness* para o repositório WEBBASE próximo ao melhor que pode ser obtido por políticas que respeitam a restrição de *politeness*.

Duas formulações que fornecem limites superiores para o *freshness* do repositório são apresentadas na Seção 4.3. Uma delas não considera a restrição de *politeness*, e a outra considera esta restrição indiretamente. Os limites superiores resultantes são chamadas UNPOLITE e POLITE, respectivamente. A solução ótima da formulação do limite superior POLITE fornece uma alocação de recursos, chamada OPT_POLITE. Propomos na Seção 4.4 um algoritmo para a alocação de recursos OPT_POLITE, cujo tempo de execução é polinomial no número de páginas e de servidores e não depende do número de revisitações.

Finalmente, a Seção 4.5 apresenta os resultados experimentais da aplicação da política DELAYED para atualizar o repositório WEBBASE, bem como os limites superiores POLITE e UNPOLITE para o *freshness* deste repositório. Os experimentos mostram grande impacto da restrição de

politeness sobre o *freshness* do repositório, que pode ser observado através da diferença entre os limites superiores POLITE e UNPOLITE. Esta diferença é válida pelo fato do limite superior UNPOLITE ser justo, visto que é atingido pela política igualmente espaçada por página. Podemos observar também que a alocação de recursos OPT_POLITE apresenta melhores resultados que a alocação de recursos proposta em (Cho03a), que não considera a restrição de *politeness*.

4.1 Política de Tempo Igualmente Espaçada por Página

A política de tempo *igualmente espaçada por página* estabelece que os intervalos entre revisitações consecutivas de uma página têm a mesma duração. Ou seja, se uma página i é atualizada com frequência f_i e recebeu uma atualização no instante t, então a próxima atualização da página i será no instante $t + f_i^{-1}$.

Definição 4.1 Política de Tempo Igualmente Espaçada por Página: *para cada página i revisitada com frequência f_i, o tempo entre revisitações consecutivas vale $1/f_i$ unidades de tempo. O instante da primeira revisitação de cada página i é uma variável aleatória uniformemente distribuída no intervalo $[0, 1/f_i)$, onde o instante 0 corresponde ao início de operação do* crawler.

A política de tempo igualmente espaçada por página e a política *fixed-order* investigada em (Cho03a) fornecem a mesma distribuição $h_i(.)$ para a duração dos intervalos entre revisitações da página i. A política *fixed-order* é apresentada na Seção 2.3.1. Conforme discutido da Seção 2.1.1, a distribuição $h_i(.)$ determina o *freshness* da página i. Portanto, a política de tempo igualmente espaçada por página fornece o mesmo *freshness* que a política *fixed-order*. De acordo com o Teorema 2.1, Cho e Garcia-Molina (Cho03a) demonstram que quando uma página i é modificada segundo um processo de Poisson com taxa λ_i, e é revisitada com frequência f_i, o *freshness* obtido pela política *fixed-order* vale

$$A_i = \frac{1 - \exp(-\lambda_i/f_i)}{\lambda_i/f_i}. \quad (4\text{-}1)$$

4.1.1 Otimalidade e Violação do Politeness

Quando não temos a restrição de *politeness*, Wolf et al. (Wol02) provam que uma estratégia para distribuir as atualizações de modo a minimizar o *staleness* (Seção 2.1.3) é mantendo-as igualmente espaçadas por página.

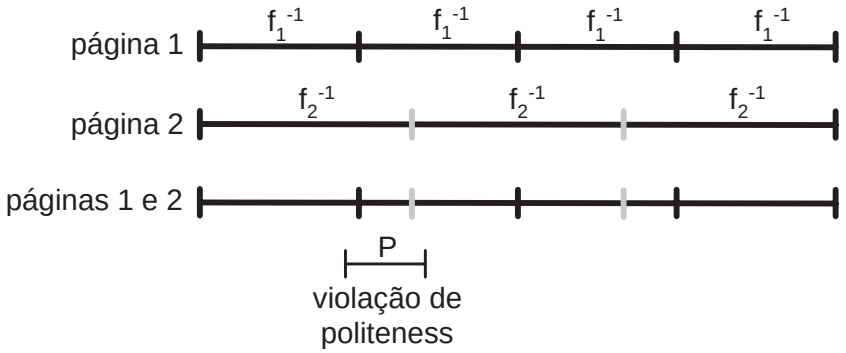

Figura 4.1: Duas páginas de um mesmo servidor com atualizações igualmente espaçadas. P é o tempo mínimo permitido entre requisições a um servidor. Pode ocorrer violação da restrição de *politeness*.

Coffman et al (Cof98) (Seção 2.3.4) também provam este resultado para uma métrica similar ao *freshness*, quando as páginas são modificadas segundo um processo de Poisson. O Lema 4.2 e o Corolário 4.3 demonstram que este resultado também é válido para o *freshness*.

Lema 4.2 *A expressão*

$$A_i^* = \frac{1 - \exp(-\lambda_i/f_i)}{\lambda_i/f_i} \tag{4-2}$$

é um limite superior para o freshness *de uma página i que é modificada segundo um processo de Poisson com frequência $\lambda_i > 0$, e é revisitada com frequência $f_i > 0$.*

Prova. Pela Equação (2-3) temos que o *freshness* A_i de uma página i atualizada com frequência f_i pode ser obtido pela expressão $A_i = f_i \times \mathrm{E}[B_i(U_i)]$, onde U_i é uma variável aleatória com distribuição de probabilidade igual à distribuição das durações dos intervalos entre revisitações à página i, e $B_i(x) = (1 - \exp(-\lambda_i x))/\lambda_i$ é o tempo esperado que a página i permanece atualizada em um intervalo entre revisitações consecutivas com duração $x > 0$ (Equação (2-2)). Como $B_i(x)$ é uma função côncava, podemos aplicar a desigualdade de Jensen (Rud87) na Equação (2-3), obtendo

$$\frac{A_i}{f_i} = \mathrm{E}[B_i(U_i)] \leq B_i(\mathrm{E}[U_i]) = B_i(f_i^{-1}) = \frac{1 - \exp(-\lambda_i/f_i)}{\lambda_i}.$$

■

Corolário 4.3 *Para o problema sem a restrição de* politeness*, a política de tempo igualmente espaçada por página é ótima com relação ao* freshness.

Prova. Comparando as Equações (4-1) e (4-2), temos que o *freshness* da página i fornecido pela política igualmente espaçada por página é igual ao limite superior para o *freshness* da página i. ■

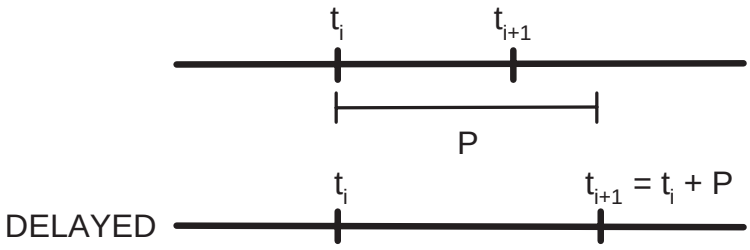

Figura 4.2: Política de tempo DELAYED: a $(i+1)$-ésima requisição ao servidor é escalonada para o instante t_{i+1} e viola a restrição de *politeness*, sendo portanto atrasada para o instante $t_i + P$, onde P é o tempo mínimo permitido entre requisições a um servidor.

Embora a política de tempo igualmente espaçada por página seja ótima em termos de *freshness*, ela pode violar a restrição de *politeness*, como ilustra a Figura 4.1. Na figura temos as páginas 1 e 2 de um mesmo servidor sendo revisitadas com frequências f_1 e f_2, respectivamente. Podemos observar na terceira linha que as duas primeiras requisições enviadas ao servidor foram muito próximas, violando a restrição de *politeness*.

Entretanto, podemos realizar ajustes nas revisitações igualmente espaçadas por página de modo que o *politeness* seja respeitado. Wolf et al. (Wol02) utilizam uma formulação baseada em problemas de transporte (Ahu93) para realizar estes ajustes, cuja solução é aproximada. Apesar desta formulação possuir algoritmo polinomial no número de revisitações, esta abordagem pode ser considerada ineficiente devido à frequência elevada de revisitações realizadas pelos *crawlers*. A próxima seção apresenta uma estratégia simples e eficiente para realizar estes ajustes, chamada DELAYED, que consiste em inserir atrasos em algumas revisitações.

4.2 Política de Tempo DELAYED

Uma estratégia para adaptar as atualizações igualmente espaçadas por página de modo a respeitar a restrição de *politeness* é inserir atrasos nas atualizações que violam esta restrição. Esta estratégia, chamada DELAYED, é ilustrada na Figura 4.2.

Definição 4.4 Política de Tempo DELAYED: *Seja P o tempo mínimo permitido entre requisições a um mesmo servidor. Para todo servidor s, se a última requisição ao servidor s ocorreu no instante t e a próxima requisição estiver escalonada pela política igualmente espaçada por página para o instante $t' < t + P$, então esta requisição será atrasada para o instante $t + P$.*

Pré-condições:

– Execução da alocação de recursos, definindo a frequência de revisitação f_i de cada página i.

Inicialização:

– Crie uma *heap* H contendo todas as páginas do repositório, onde a chave t_i de cada página i é escolhida uniformemente no intervalo $[0, f_i^{-1})$.
O topo da *heap* H contém a página com menor chave.

Operação do *crawler*:

1. Execute os passos abaixo no início da operação e sempre que uma revisitação for realizada:
   (a) Remova da *heap* H a página i com menor chave t_i.
   (b) Escalone a revisitação da página i para o instante $\max\{t_i, t_s + P\}$, onde t_s é o instante da última requisição ao servidor s que hospeda a página i.
   (c) Insira a página i na *heap* H com chave $t_i + f_i^{-1}$.

Figura 4.3: Pseudo-código da política de tempo DELAYED.

O pseudo-código da política DELAYED é apresentado na Figura 4.3. Uma *heap* é utilizada para determinar de forma eficiente a próxima página a ser revisitada. Como a página revisitada retorna para a *heap* no passo 1(c) com um incremento de f_i^{-1} em sua chave, temos que a chave de cada página i na *heap* representa o instante da próxima revisitação de i quando utilizamos a política de tempo igualmente espaçada por página. Entretanto, a política DELAYED não escalona no passo 1(b) a revisitação para o instante indicado pela chave t_i da página i com menor chave, mas para o instante $\max\{t_i, t_s+P\}$, para evitar a violação da restrição de *politeness*. A inicialização cria uma *heap* contendo todas as páginas, e portanto tem complexidade de tempo da ordem do número de páginas no repositório. Devido ao reposicionamento da página na *heap*, o escalonamento de cada revisitação tem tempo de execução da ordem do logaritmo do número de páginas.

4.3 Limites Superiores para o Freshness do Repositório

Seja $A_i^*(f_i)$ o limite superior fornecido pelo Lema 4.2 para o *freshness* da página i quando revisitada com frequência f_i. Então, $\sum_{i \in R} w_i A_i^*(f_i)$ é um limite superior para o *freshness* do repositório R para dada uma atribuição de frequência f_i para cada página i, onde w_i é a importância da página i. Portanto, para determinar um limite superior para o *freshness* do repositório

R para qualquer atribuição de frequências, determinamos as frequências não negativas que maximizam $\sum_{i \in R} w_i A_i^*(f_i)$ sujeitas à $\sum_{i \in R} f_i = C$, onde C é a frequência total de revisitações realizada pelo *crawler*.

Definição 4.5 *O* **Limite Superior UNPOLITE** *para o* freshness *de um repositório R é o valor ótimo da formulação*

$$\begin{aligned} \textit{maximize} \quad & \sum_{i \in R} w_i A_i^*(f_i) \\ \textit{sujeito à} \quad & \sum_{i \in R} f_i = C, \\ & f_i \geq 0, \ \textit{para toda página } i \in R, \end{aligned}$$

onde w_i é a importância da página i, C é a frequência total de revisitações realizadas pela crawler*, e $A_i^*(f_i)$ é o limite superior fornecido pelo Lema 4.2 para o* freshness *da página i atualizada com frequência f_i.*

Conforme discutido na Seção 4.1.1, este limite superior pode ser alcançado pela política de tempo igualmente espaçada por página, porém o escalonamento resultante pode violar a restrição de *politeness*. Cho e Garcia-Molina (Cho03a) mostram que uma solução para a formulação do limite superior UNPOLITE pode ser obtida em tempo $O(|R|)$ através da relaxação lagrangiana da restrição $\sum_{i \in R} f_i = C$, cuja solução ótima é chamada aqui de alocação de recursos CHO.

Definição 4.6 *A* **alocação de recursos CHO** *é a alocação fornecida pela solução ótima da formulação do limite superior UNPOLITE.*

Como o limite superior UNPOLITE não considera a restrição de *politeness*, este limite pode estar muito acima do maior *freshness* alcançado por políticas que respeitam a restrição de *politeness*. Propomos então um conjunto de restrições para as frequências de revisitação de modo a produzir um limite mais justo: como um tempo mínimo P deve ser respeitado entre requisições consecutivas a um servidor s, então a soma das frequências de revisitação das páginas hospedadas no servidor s deve ser no máximo $1/P$. A formulação deste novo limite superior é apresentada na Definição 4.7, onde as restrições adicionais aparecem na Equação (4-4).

Definição 4.7 *O* **Limite Superior POLITE** *para o* freshness *do repositório R é o valor ótimo do problema*

$$\mathcal{P} : \textit{maximize} \quad \sum_{i \in R} w_i A_i^*(f_i)$$

$$\textit{sujeito à} \quad \sum_{i \in R} f_i = C, \tag{4-3}$$

$$\sum_{i \in R_s} f_i \leq P^{-1}, \ \textit{para todo servidor } s, \tag{4-4}$$

$$f_i \geq 0, \ \textit{para toda página } i \in R,$$

onde R_s é o conjunto de páginas no servidor s, w_i é a importância da página i, C é a frequência total de revisitações realizadas pelo crawler*, $A_i^*(f_i)$ é o limite superior fornecido pelo Lema 4.2 para o* freshness *da página i atualizada com frequência f_i, e P é o tempo mínimo permitido entre requisições a um servidor. A frequência total C não pode ser maior que mP^{-1}, onde m é o número de servidores.*

4.4 Alocação de Recursos considerando a Restrição de Politeness

A solução ótima da formulação do limite superior POLITE fornece uma alocação de recursos, chamada OPT_POLITE. A alocação de recursos OPT_POLITE pode apresentar melhores resultados que a alocação de recursos CHO, pois considera a restrição de *politeness*. Estas duas alocações de recursos são comparadas experimentalmente na Seção 4.5. Esta seção apresenta um algoritmo eficiente para a alocação de recursos OPT_POLITE. Para simplificar a apresentação assumimos que as páginas possuem importâncias uniformes. Os resultados podem ser generalizados para importâncias não uniformes.

Definição 4.8 *A* **alocação de recursos OPT_POLITE** *é a alocação fornecida pela solução ótima da formulação do limite superior POLITE.*

Segundo a classificação apresentada em (Bre02), a formulação do limite superior POLITE é um problema da mochila contínuo, não linear, convexo, separável, e com restrições adicionais do tipo *block diagonal*. Note que aplicando a relaxação lagrangiana à restrição (4-3) obtemos m problemas da mochila independentes, onde m é o número de servidores. Segundo (Bre02), os algoritmos propostos para este tipo de problema tipicamente operam mantendo um conjunto de condições de primeira ordem (Bor02) satisfeitas, e aplicam esforço computacional para incrementar este conjunto até que todas as condições de primeira ordem sejam satisfeitas. Um algoritmo desta categoria para a alocação de recursos OPT_POLITE é apresentado na Figura 4.4. Em cada iteração do passo 2 encontramos as frequências ótimas para a formulação sem as restrições (4-4). Estas frequências ótimas podem ser obtidas utilizando o algoritmo proposto em (Cho03a) para a alocação de recursos CHO, com tempo de execução da ordem do número de variáveis. Em seguida, identificamos o

1. Inicialização: $S^{(0)} \leftarrow \emptyset, \quad k \leftarrow 0.$

2. Faça,

   (a) Seja $N^{(k)}$ o conjunto de páginas do repositório que não estão hospedadas em servidores contidos em $S^{(k)}$.
   Encontre as frequências ótimas f_i^* para a formulação abaixo, utilizando o algoritmo proposto em (Cho03a).

$$\begin{aligned} \mathcal{U}(S^{(k)}) : \text{maximize} \quad & \sum_{i \in N^{(k)}} A_i^*(f_i) \\ \text{sujeito à} \quad & \sum_{i \in N^{(k)}} f_i = C - |S^{(k)}| \times P^{-1}, \\ & f_i \geq 0, \text{ para toda página } i \in N^{(k)}. \end{aligned}$$

   (b) Seja V o conjunto de servidores $s \notin S^{(k)}$ tal que $\sum_{i \in R_s} f_i^* > P^{-1}$, onde R_s é o conjunto de páginas hospedadas em s.
   Para cada servidor $s \in V$, encontre as frequências ótimas para as páginas de s através da formulação abaixo, utilizando o algoritmo proposto em (Cho03a).

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in R_s} A_i^*(f_i) \\ \text{sujeito à} \quad & \sum_{i \in R_s} f_i = P^{-1}, \\ & f_i \geq 0, \text{ para toda página } i \in R_s. \end{aligned}$$

   (c) $S^{(k+1)} \leftarrow S^{(k)} \cup V.$

   (d) $k \leftarrow k + 1.$

   Enquanto $V \neq \emptyset$.

Figura 4.4: Algoritmo para a alocação de recursos OPT_POLITE.

conjunto V de servidores onde a restrição (4-4) é violada por estas frequências ótimas. Fixamos então em P^{-1} a soma das frequências das páginas hospedadas nos servidores em V. Com isso podemos aplicar novamente o algoritmo da alocação de recursos CHO para encontrar as frequências ótimas das páginas de cada servidor em V. As páginas dos servidores em V podem então ser removidas do modelo, e executamos outra iteração do passo 2. Este processo é repetido até que nenhum servidor que permaneceu no modelo tenha a restrição (4-4) violada.

Como em cada iteração k pelo menos um servidor entra no conjunto $S^{(k)}$, e em cada iteração o algoritmo para alocação de recursos CHO é executado duas vezes, no pior caso o algoritmo para alocação de recursos CHO é executado

$2m$ vezes, onde m é o número se servidores. Cada execução do algoritmo alocação de recursos CHO tem tempo de execução da ordem do número de página n do repositório. Portanto, o algoritmo para alocação de recursos OPT_POLITE tem tempo de execução $O(nm)$.

Lema 4.9 *O algoritmo apresentado na Figura 4.4 fornece uma solução ótima para a formulação do limite superior POLITE, quando consideramos importâncias uniformes.*

Prova. Seja S^* um conjunto de servidores onde a restrição (4-4) está ativa em uma solução ótima $f_i^*, \forall i \in R$, para a alocação de recursos OPT_POLITE. Ou seja, $\sum_{i \in R_s} f_i^* = P^{-1}$ para todo servidor $s \in S^*$. Conhecendo o conjunto S^*, podemos encontrar as frequências ótimas das páginas dos servidores em S^* com o algoritmo de (Cho03a), e fixar estas frequências no modelo original. O modelo resultante pode então ser resolvido utilizando o algoritmo de (Cho03a). Portanto, como $S^{(0)} \subseteq S^*$, para provar o teorema basta mostrar que se $S^{(k)} \subset S^*$ então $S^{(k+1)} \subseteq S^*$. A prova apresentada a seguir é feita por contradição, onde $S^{(k)} \subset S^*$ e $S^{(k+1)} \nsubseteq S^*$ implicam na existência de um servidor s onde a restrição (4-4) está ativa na solução ótima, mas não está ativa na solução fornecida pela alocação de recursos OPT_POLITE. Mostramos então que é possível melhorar esta solução ótima atribuindo uma frequência maior às páginas de s.

Seja s um servidor tal que a solução ótima de $\mathcal{U}(S^{(k)})$ viola a restrição (4-4), mas não ocorre em S^*. Neste caso podemos concluir que $q_s^* < P^{-1} < q_s^{(k)}$, onde $q_s^{(k)}$ denota a frequência total destinada ao servidor s na solução ótima de $\mathcal{U}(S^{(k)})$, e q_s^* denota a frequência total destinada ao servidor s na alocação de recursos OPT_POLITE. Como $S^{(k)} \subset S^*$ e a alocação de recursos OPT_POLITE respeita as restrições (4-3) e (4-4), então existe um servidor $r \notin S^{(k)}$ tal que $q_r^{(k)} < q_r^* \leq P^{-1}$.

De acordo com as condições de primeira ordem do problema $\mathcal{U}(S^{(k)})$, para cada página i não hospedada nos servidores em $S^{(k)}$, as derivadas $\partial A_i^*(f_i)/\partial f_i$ são todas iguais em uma solução ótima. Além disso, como $A_i^*(f_i)$ é uma função côncava (derivada de segunda ordem negativa) e crescente com relação à f_i, então a parcela $\sum_{i \in R_s} A_i^*(f_i)$ da função objetivo de $\mathcal{U}(S)$ correspondente às páginas do servidor s é uma função côncava e crescente com relação à frequência total $q_s = \sum_{i \in R_s} f_i$ de revisitação das páginas de s. Isto implica que para valores maiores de q_s, menor será o ganho na função objetivo ao aumentar em $\delta > 0$ o valor de q_s. Portanto, podemos melhorar a solução ótima f_i^*, $\forall i \in R$, transferindo $0 < \delta \leq \min\{P^{-1} - q_s^*,\ q_r^* - q_r^{(k)}\}$ unidades de frequência do servidor r para o servidor s, pois a redução na função objetivo provocada

pela redução de δ em q_r^* é menor que o ganho provocado pelo aumento de δ em $q_s^{(k)}$. ■

4.5 Resultados Experimentais

A Figura 4.5 apresenta o *freshness* do repositório WEBBASE durante a simulação de quatro anos de operação do *crawler*. As páginas que apresentaram modificação durante o período de monitoramento do repositório WEBBASE são consideradas obsoletas no início da simulação. As páginas restantes são consideradas sempre atualizadas. Duas políticas de revisitação são avaliadas nesta simulação: a política de tempo DELAYED com alocação de recursos CHO, e a política de tempo DELAYED com alocação de recursos OPT_POLITE. Os limites superiores UNPOLITE e POLITE para o *freshness* do repositório WEBBASE também aparecem nos gráficos da Figura 4.5. A frequência de requisições a um servidor é no máximo o inverso do tempo mínimo P permitido entre requisições a um servidor, e portanto a frequência total C de revisitações realizadas pelo *crawler* é no máximo P^{-1} vezes o número de servidores. Nos gráficos da Figura 4.5, a frequência total C assume 1%, 10% e 90% desta frequência total máxima. O tempo mínimo P permitido entre requisições a um mesmo servidor foi fixado em 15 segundos.

Podemos observar nos gráficos na Figura 4.5 que:

- A diferença entre os limites superiores UNPOLITE e POLITE cresce com o aumento da frequência total de revisitação C. Esta diferença chega a quase 20% para C igual a 90% da frequência total máxima, o que indica grande impacto da restrição de *politeness*. Esta diferença é válida pelo fato do limite superior UNPOLITE ser justo, visto que é atingido pela política igualmente espaçada por página.
- O *freshness* da política de tempo DELAYED com alocação de recursos OPT_POLITE se aproxima muito do limite superior POLITE após 4 anos de operação do *crawler*, com uma diferença inferior a 2,4% de *freshness* do repositório nos três gráficos. Isto sugere que (i) esta política simples e eficiente deixa pouca margem de melhora do *freshness* do repositório para políticas mais sofisticadas, e (ii) o limite superior POLITE é muito próximo ao limite real alcançável por políticas que respeitam o *politeness*.
- A política DELAYED apresenta melhores resultados quando utiliza a alocação de recursos OPT_POLITE, e a diferença para a alocação de recursos CHO cresce com o aumento da frequência total C, chegando a 2,8% para C igual a 90% da frequência total máxima.

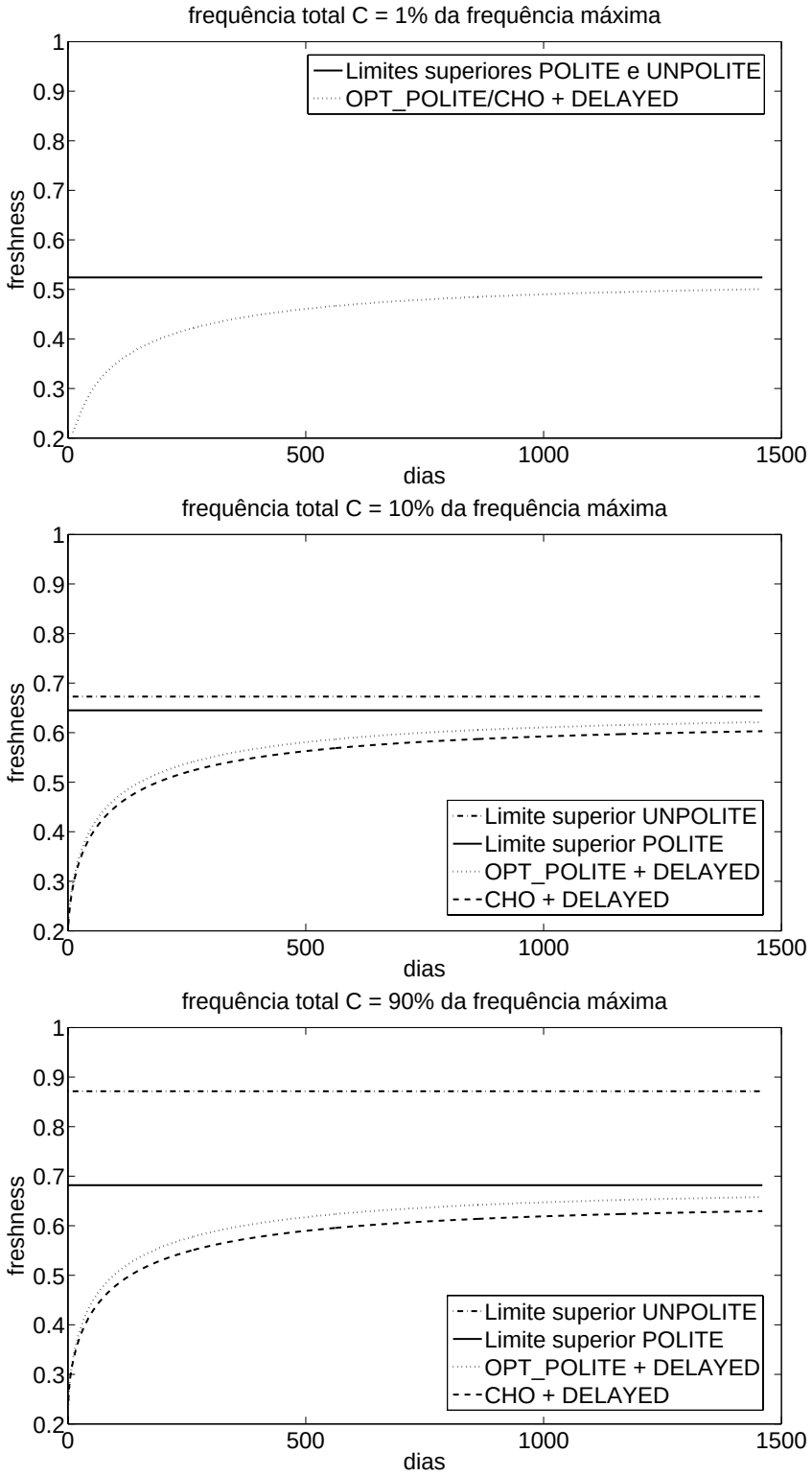

Figura 4.5: *Freshness* do repositório WEBBASE fornecido pela política DELAYED durante 4 anos de operação do *crawler*, para a frequência total C de revisitação igual a 1%, 10% e 90% da frequência máxima permitida pela restrição de *politeness*.