Tractable near-optimal policies for crawling

Yossi Azar[a], Eric Horvitz[b], Eyal Lubetzky[c], Yuval Peres[b,1], and Dafna Shahaf[d]

[a]Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel; [b]Microsoft Research, Redmond, WA 98052; [c]Courant Institute of Mathematical Sciences, New York University, New York, NY 10012; and [d]School of Computer Science, Hebrew University, Jerusalem 91904, Israel

Contributed by Yuval Peres, May 8, 2018 (sent for review March 30, 2018; reviewed by Anthony Bonato and Robert D. Kleinberg)

The problem of maintaining a local cache of *n* constantly changing pages arises in multiple mechanisms such as web crawlers and proxy servers. In these, the resources for polling pages for possible updates are typically limited. The goal is to devise a polling and fetching policy that maximizes the utility of served pages that are up to date. Cho and Garcia-Molina [(2003) *ACM Trans Database Syst* 28:390–426] formulated this as an optimization problem, which can be solved numerically for small values of *n*, but appears intractable in general. Here, we show that the optimal randomized policy can be found exactly in *O*(*n* log *n*) operations. Moreover, using the optimal probabilities to define in linear time a deterministic schedule yields a tractable policy that in experiments attains 99% of the optimum.

web crawling | caching policies | scheduling optimization

Modern web crawlers and proxy servers strive to maintain an up-to-date mirror of the Internet. More generally, various real-world applications rely on local caches of resources that are changing without notification. This typically necessitates actively polling the pages for changes, thus raising the optimization problem of prioritizing these polls.

Our goal is to maximize the throughput of up-to-date pages served, whence the two dominant factors in the optimization are the change rate of the pages and their utility (for example, as represented by the frequency of requests for the pages). Intuitively, the crawler should keep important pages as fresh as possible. However, combining these criteria is nontrivial: For instance, the server may wish to prioritize pages that are frequently updated and, at the same time, ones that are more likely to be requested in the future. Interestingly, the server might want to avoid polling a page even if both criteria are met [e.g., if its change rate is too high (since its local copy would quickly be outdated and thus have lower utility)].

We focus here on the problem of maintaining a cache of n pages for which freshness is imperative, as formalized in the pioneering work of Cho and Garcia-Molina (ref. 1) via a binary classification of freshness (see also ref. 2). This could fit, e.g., a small cache of pages that, if served, are guaranteed to be genuinely up to date, whereas other requests would be served via slower mechanisms. In ref. 1, the corresponding optimization problem was numerically solved for small values of n in the special case where all pages have equal popularity (i.e., each request is uniform over the set of pages). The pages change via Poisson processes, and the authors rely on a separate work (ref. 3) to estimate their change rates from historical data. It was shown in ref. 3 (following ref. 4) that, under the assumption that all pages have the same importance, the optimal policy is neither the uniform one nor proportional to the change rates, since "to improve freshness, we should penalize the elements that change too often." Rather, as described in the account of that work in ref. 5:

> The optimal method for keeping average freshness high includes ignoring the pages that change too often, and the optimal for keeping average age low is to use access frequencies that monotonically (and sublinearly) increase with the rate of change of each page. ...Explicit formulas for the re-visit policy are not attainable in general, but they are obtained numerically, as they depend on the distribution of page changes. Note that the re-visiting policies considered here regard all pages as homogeneous in terms of quality – all pages on the Web are worth the same – something that is not a realistic scenario, so further information about the Web page quality should be included to achieve a better crawling policy.

The more challenging setting where the page requests are nonuniform is mentioned in ref. 1, section 6, where the authors discuss the case of two possible weights for popularity.

Here, we provide an efficient solution for the general case of arbitrary utility values and change frequencies for the n pages. First, we show that the optimal randomized policy—whereby each page is assigned a rate at which it is crawled, independently of the other pages, subject to a constraint in terms of the overall bandwidth—can be recovered exactly (as opposed to numerically solved) in near-linear time. [Here, and in what follows, we refer to a "randomized policy" as a shorthand for a (stationary) policy where the pages to be refreshed are drawn independently from the same distribution.] Thereafter, this solution yields, in linear time, a deterministic (cyclic) policy maintaining the same update frequencies (this typically outperforms the random policy one, as discussed in ref. 1), which in numerical experiments achieved 99% of the (numerically solved) optimal solution (see Fig. 1 for its performance on synthetic data in the framework of ref. 1). These results extend via the same method to other flavors of the model, such as discrete-time policies (where time is divided into slots, in each of which the server polls a single page for an update), as well as to situations where the freshness of each page is continuous, and its decay is modeled by an exponential random variable.

We stress that one may incorporate mechanisms to approximate the change rates (as in ref. 3) and utilities of pages so as to continually update the crawling policies as per *Theorem 1* (see, e.g., refs. 8–10, as well as refs. 11 and 12 for additional related work on policies for web crawling).

Significance

We present a tractable algorithm that provides a near-optimal solution to the crawling problem, a fundamental challenge at the heart of web search: Given a large quantity of distributed and dynamic web content, what pages do we choose to update a local cache with the goal of serving up-to-date pages to client requests? Solving this optimization requires identifying the best set of pages to refresh given popularity rates and change rates—an intractable problem in the general case. To overcome this intractability, we show that the optimal randomized strategy can be efficiently determined (in near-linear time) and then use it to produce a deterministic policy that exhibits excellent performance in experiments.

Author contributions: E.H. designed research; Y.A., E.H., E.L., Y.P., and D.S. performed research; and E.L. and Y.P. wrote the paper.

Reviewers: A.B., Ryerson University; and R.D.K., Cornell University.

The authors declare no conflict of interest. Reviewer R.D.K. is a former employee of Microsoft Research and has been a consultant for the company.

[1]To whom correspondence should be addressed. Email: peres@microsoft.com.

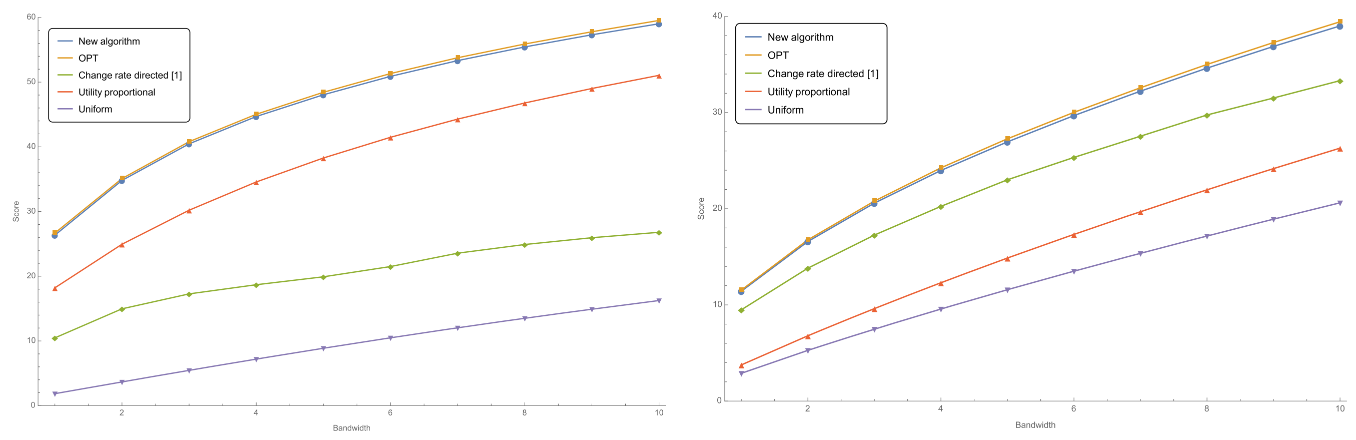

Fig. 1. Performance of the new algorithm for *Problem 2* (solved exactly) vs. the optimum (OPT, numerically solved), the change rate-directed policy studied in ref. 1 (numerically solved), and utility-proportional and random (uniform) policies. A synthetic dataset had 1,000 pages with change rates independent and identically distributed (i.i.d.) Uniform [0, 1], and utilities were (independently) either i.i.d. Zipf (with exponent 2 and range 10000) or i.i.d. Uniform [0, 1]. For more on the standard modeling of web page utilities via power laws see, e.g., refs. 6 and 7, section 2.

Beyond the scope of web crawling, an efficient solution for the above problem may find other applications in the context of bicriteria optimization and scheduling problems. A related problem of scheduling maintenance service to machines was studied in refs. 13 and 14, where the authors assumed activities of several types, under the constraint that at most a single activity can be scheduled to any one period. The problem is then to find an optimal schedule specifying at which periods to execute each of the activity types to minimize the long-run average cost per period.

Optimal Randomized Update Policy

We formalize the optimization problem for finding a random (stationary) policy first for the discrete-time setting, followed by the continuous-time setting which was studied in ref. 1.

Discrete-Time Policies. A server maintains a pool of n pages. Time is partitioned into units, to be thought of as the amount of time it takes the server to update a single page in its pool. We use the following two standard ways to quantify the average freshness of these pages over time, weighted by their request frequency or importance (see, e.g., ref. 1, definitions. 2.1 and 6.1). In this model, the server updates its pages in discrete time, while requests may occur in continuous time.

i) Request rate model: Requests for pages arrive according to independent Poisson processes: Let $\mu_i > 0$ denote the rate of requests for page i. Formally, the (random) request sequence is $\{(I_j, T_j)\}_{j=1}^N$, where (I_j, T_j) denotes a request at time T_j for page $I_j \in \{1, \ldots, n\}$, and $0 = T_0 < T_1 < T_2 < \ldots$; let $\text{FRESH}(i, t)$ be the event that page i is fresh at time t. The goal of the server is to maximize

$$\mathbb{E}\left[\frac{1}{N}\sum_{j=1}^{N}\mathbf{1}\{\text{FRESH}(I_j, T_j)\}\right]. \qquad [1]$$

[See Fig. 1 for simulations depicting $\sum_{j=1}^{N}\mathbf{1}\{\text{FRESH}(I_j, T_j)\}$ for various update policies.]

ii) Utility model: Each page is assigned a predetermined nonnegative weight μ_i measuring its importance. Given the time horizon H, the server wishes to maximize

$$\mathbb{E}\left[\frac{1}{H}\sum_{t=1}^{H}\sum_{i=1}^{n}\mu_i\,\mathbf{1}\{\text{FRESH}(i, t)\}\right]. \qquad [2]$$

In both models, pages are modified in the following way: Let $0 < \Delta_i < 1$ denote the probability that page i is changed in a given time unit, independently of other pages and time steps.

The update policy of the server comprises a distribution $\{p_i\}_{i=1}^n$ over pages. For concreteness, assume that in each time unit the server updates one page independently via this distribution (alternatively, the server may construct a periodic scheduling policy using these frequencies; see *Algorithm 3*). If page i is changed and the server updates it in the same time step, then this page is fresh until the next time it changes.

We will argue that, for large N and H, both models correspond to the following:

Problem 1 (Discrete-Time Randomized Policy). INPUT: *Utility and change rates for each of the pages:* $\{(\mu_i, \Delta_i)\}_{i=1}^n$ *(where* $\mu_i > 0$ *and* $0 < \Delta_i < 1$ *for all* i*).*

OUTPUT: *Update frequencies for the pages:* $\mathbf{p} = (p_1, \ldots, p_n)$ *(where* $p_i \geq 0$ *for all* i *and* $\sum_i p_i = 1$*), maximizing*

$$F_1(\mathbf{p}) = \sum_i \frac{\mu_i p_i}{p_i + \Delta_i - p_i\Delta_i}. \qquad [3]$$

Continuous-Time Policy. The following was studied in ref. 1 for the case where $\mu_i = 1$ for all i.

iii) Request and change rates model: Let $\Delta_i > 0$ denote a Poisson rate at which page i is changed, and let $\mu_i > 0$ denote the Poisson rate at which page i is being requested (part a). Let $\rho_i \geq 0$ denote the Poisson rate at which the server updates page i, where $\sum \rho_i = R$ for some $R > 0$ (the total bandwidth). Given R and some time horizon H, the goal of the server is to maximize

$$\mathbb{E}\left[\frac{1}{H}\int_0^H\left(\sum_{j=1}^{n}\mu_j\mathbf{1}\{\text{FRESH}(j, t)\}\right)dt\right]. \qquad [4]$$

(cf. ref. 1, theorem 4.1). We will argue that, for large H, this optimal random policy corresponds to the following problem:

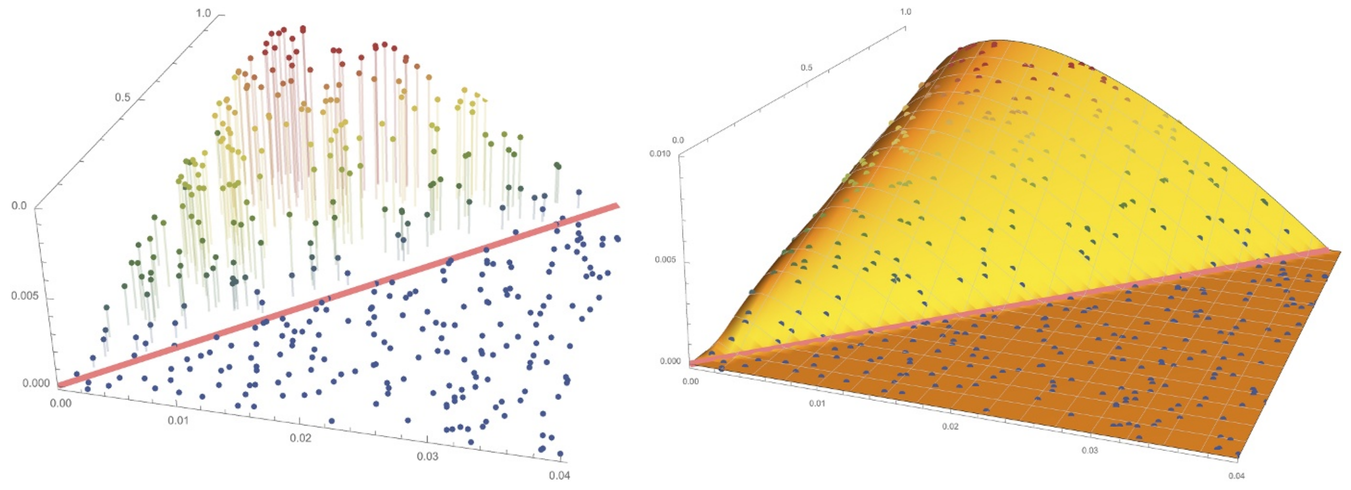

Fig. 2. Optimal solution **p*** [global minimum of $F_1(\mathbf{p})$] for a synthetic dataset. In red, the line $\mu = \lambda\Delta$ is representing the threshold λ such that $p_i{}^* = 0$ if and only if $\mu_i/\Delta_i \leq \lambda$.

Problem 2 (Continuous-Time Randomized Policy). INPUT: *Bandwidth* $R > 0$ *and request and change rates for each of the pages:* $\{\mu_i, \Delta_i\}_{i=1}^n$ *(where* $\mu_i, \Delta_i > 0$ *for all* i*).*

OUTPUT: *Update rates for the pages:* $\rho = (\rho_1, \ldots, \rho_n)$ *(where* $\rho_i \geq 0$ *for all* i *and* $\sum_i \rho_i = R$*), maximizing*

$$F_2(\rho) = \sum_i \frac{\mu_i \rho_i}{\rho_i + \Delta_i}. \qquad [5]$$

Note that the cost function $F_2(\rho)$ above is the cost function for a random stationary policy; in the context of a deterministic policy where the update rates are ρ_i, one would replace it by

$$\overline{F}(\rho) = \sum_i \mu_i \frac{1 - \exp(-\Delta_i/\rho_i)}{\Delta_i/\rho_i}, \qquad [6]$$

as in ref. 1, section 4 (denoted the average freshness of the database).

Solution. Our main result provides efficient algorithms to find the unique solutions for *Problems 1* and *2*.

Theorem 1. *For each set of input parameters for Problem 1, there is a unique* $\mathbf{p}^*$ *that achieves the global maximum of* $F_1(\mathbf{p})$ *subject to the constraints* $0 \leq p_i \leq 1$ *and* $\sum_i p_i = 1$*. Furthermore, there is an explicit algorithm (Algorithm 1 below) that finds the unique solution* $\mathbf{p}^*$ *in time* $O(n \log n)$*.*

Analogously, for Problem 2, for every set of admissible input parameters, there exists a unique ρ^* *achieving the global maximum of* $F_2(\rho)$*, and Algorithm 2 finds the solution* ρ^* *in time* $O(n \log n)$*.*

Following are the aforementioned algorithms for finding the optimal policies $\mathbf{p}^*$:

Algorithm 1: Find Optimum of Problem 1

```
Input: $\mu, \Delta$ popularity and change rate vectors of length $n$
Output: p stochastic policy
1. Sort $\mu, \Delta$ by an ascending order of $\mu_i/\Delta_i$.
2. Let
   $r \hookleftarrow \sum_{i=1}^n \frac{\sqrt{\mu_i\Delta_i}}{1-\Delta_i}, \quad s \hookleftarrow \sum_{i=1}^n \frac{\Delta_i}{1-\Delta_i}.$
3. for $i = 1$ to $n$ do
4. if $\mu_i/\Delta_i \leq \left(\frac{r}{1+s}\right)^2$ then
5. $p_i \hookleftarrow 0$
6. $r \hookleftarrow r - \frac{\sqrt{\mu_i\Delta_i}}{1-\Delta_i}$
7. $s \hookleftarrow s - \frac{\Delta_i}{1-\Delta_i}$
8. else
9. $p_i \hookleftarrow \frac{(1+s)\sqrt{\mu_i\Delta_i} - r\Delta_i}{r(1-\Delta_i)}$
10. Return p.
```

Algorithm 2: Find Optimum of Problem 2

```
Input: Bandwidth $R > 0$ and $\mu, \Delta$ popularity and change rate vectors of length $n$.
Output: $\rho$ stochastic policy
1. Sort $\mu, \Delta$ by an ascending order of $\mu_i/\Delta_i$.
2. Let
   $r \hookleftarrow \sum_{i=1}^n \sqrt{\mu_i\Delta_i}, \quad s \hookleftarrow \sum_{i=1}^n \Delta_i.$
3. for $i = 1$ to $n$ do
4. if $\mu_i/\Delta_i \leq \left(\frac{r}{R+s}\right)^2$ then
5. $\rho_i \hookleftarrow 0$
6. $r \hookleftarrow r - \sqrt{\mu_i\Delta_i}$
7. $s \hookleftarrow s - \Delta_i$
8. else
9. $\rho_i \hookleftarrow \sqrt{\mu_i\Delta_i}(R+s)/r - \Delta_i$
10. Return. $\rho$
```

Both algorithms proceed by first sorting the pages according to μ_i/Δ_i, then linearly processing them to determine $\mathbf{p}^*$ (Fig. 2).

Proofs. To see that *Problems 1* and *2* indeed capture the objective functions in the aforementioned models, first consider the request rate model. Since requests for page i arrive independently according to a Poisson (μ_i) process, given the history until time T_{j-1}, the variable $T_j - T_{j-1}$ is a minimum of independent exponential variables with rates $\mu_1, \ldots, \mu_n$, and

$$\mathbb{P}(I_j = i) = \mu_i / \sum_l \mu_l \quad \text{for every } i \in \{1, \ldots, n\}. \qquad [7]$$

Recall that the following holds for any t_0. Page i was initially outdated at time 0, and in each of the time units $t \in \{1, \ldots, t_0\}$ it had probability p_i to be fetched (denote this event as B_t) and probability Δ_i to be modified (denote this event as C_t), both events being independent of each other [hence $\mathbb{P}(B_t \cap C_t) = p_i\Delta_i$] and other time steps. Page i is fresh at time $t_0 + 1$ if and only if the event B_t occurred for some $t \le t_0$, and C_t did not occur between times $t+1, \dots, t_0$. In particular, if we let

$$E_{t_0} := \bigcup_{l=1}^{t_0} (B_l \cup C_l),$$

(the event that page i is either fetched or modified before time t_0+1), then $E_{t_0} \supset \text{FRESH}(i, t_0+1)0$, and since $\mathbb{P}(\text{FRESH}(i, t_0+1) \mid E_{t_0}) = \frac{p_i}{p_i+\Delta_i - p_i\Delta_i}$ (by looking at the last time point l where $B_l \cup C_l$ occurred), it now follows that

$$\begin{aligned}\mathbb{P}(\text{FRESH}(i, t_0+1)) &= \mathbb{P}(\text{FRESH}(i, t_0+1) \mid E_{t_0})\mathbb{P}(E_{t_0}) \\ &= \frac{p_i}{p_i+\Delta_i-p_i\Delta_i}\left[1-((1-p_i)(1-\Delta_i))^{t_0}\right].\end{aligned}$$

Let $\epsilon > 0$. It now suffices to take a burn-in period logarithmic in ϵ (recall that every Δ_i is bounded away from 1), namely, $t_0 = \lceil \log_{1-\delta} \epsilon \rceil$ where $\delta = \min_i \Delta_i$. For all $T_j > t_0$ we have that $\mathbb{P}(\text{FRESH}(I_j, T_j))$ is simply

$$\sum_i \frac{\mu_i}{\sum_l \mu_l} \cdot \frac{p_i}{p_i+\Delta_i-p_i\Delta_i} = \frac{1}{\sum_l \mu_l} F_1(\mathbf{p}),$$

up to a multiplicative error of $1+\epsilon$. This matches Eq. **1** up to the policy-independent factor $\sum_l \mu_l$, as required.

An alternative derivation of $F_1(\mathbf{p})$ in the request rate model is to write an explicit form for $x_t^{(i)} = \mathbb{P}(\text{FRESH}(i,t))$. At time 0 we have $x_0^{(i)} = 0$ for all i, and for $t \ge 1$ we have

$$x_{t+1}^{(i)} = (1-p_i)(1-\Delta_i)x_t^{(i)} + p_i.$$

Iterating the above recursion (and using that $x_0^{(i)} = 0$) we find that

$$\begin{aligned}x_t^{(i)} &= p_i \frac{1-((1-p_i)(1-\Delta_i))^t}{1-(1-p_i)(1-\Delta_i)} \\ &= \frac{p_i}{p_i+\Delta_i-p_i\Delta_i}\left[1-((1-p_i)(1-\Delta_i))^t\right],\end{aligned}$$

giving the same approximation guarantee.

In view of Eq. **7**, we see that the optimized quantity in the request rate model was precisely $\sum_t \sum_i \frac{\mu_i}{\sum_j \mu_j} \mathbb{P}(\text{FRESH}(i,t))$, which coincides with Eq. **2** (the utility model) up to rescaling.

Finally, consider the continuous-time model. Repeating the arguments that led to Eq. **3**, and using the fact that with probability $\frac{\rho_i}{\rho_i+\Delta_i}$ the exponential clock that rings first among two with rates ρ_i, Δ_i is the one corresponding to the page update, the objective function in the above setting now takes the form given in *Problem 2*.

When comparing Eq. **5** to Eq. **3**, the denominator lacks the term $\rho_i\Delta_i$, which corresponds to $\mathbb{P}(B_t \cap C_t)$, the probability that at time t page i would both be updated and be modified. Intuitively, in continuous time this term no longer exists, since each of these events now occurs whenever an independent exponential clock rings, hence they almost surely never occur at the same time.

Proof of Theorem 1: Recalling the definition of $F_1(\mathbf{p})$ as given in Eq. **3**, for every $1 \le i < j \le n$ we have

$$\frac{\partial F_1}{\partial p_i} = \frac{\mu_i \Delta_i}{(p_i+\Delta_i-p_i\Delta_i)^2}, \qquad \textbf{[8]}$$

$$\frac{\partial^2 F_1}{\partial p_i^2} = -\frac{2\mu_i\Delta_i(1-\Delta_i)}{(p_i+\Delta_i-p_i\Delta_i)^3}, \qquad \frac{\partial^2 F_1}{\partial p_i \partial p_j} = 0.$$

Let $\mathcal{Q} = [0,1]^n$. Using Lagrange multipliers, if $\mathbf{p} = (p_1, \dots, p_n) \in \mathcal{Q}$ is a local maximum of F_1 and satisfies $\sum_i p_i = 1$, then either it belong to $\partial\mathcal{Q}$, the boundary of the domain $\mathcal{Q}$, or $\mathbf{p}$ is a solution to the following system of equations in $p_1, \dots, p_n$, $\lambda \in \mathbb{R}$:

$$\lambda = \frac{\mu_i\Delta_i}{(p_i+\Delta_i-p_i\Delta_i)^2} \qquad (i = 1, \dots, n),$$

$$\sum_{i=1}^n p_i = 1.$$

[Note that the constraint function $g(\mathbf{p}) := \sum_i p_i$ has $\nabla g = (1, \dots, 1)$; in particular, $\nabla g \neq 0$ in $\mathcal{Q}$.] Moreover, since F_1 is concave on $\mathcal{Q}$, every local extremum in the interior of $\mathcal{Q}$ is a global maximum.

Rearranging the first equation, we have

$$p_i = \frac{\sqrt{\frac{\mu_i\Delta_i}{\lambda}} - \Delta_i}{1-\Delta_i}, \qquad \textbf{[9]}$$

and plugging it in the second one it follows that

$$\lambda = \left(\frac{\sum_i \frac{\sqrt{\mu_i\Delta_i}}{1-\Delta_i}}{1+\sum_i \frac{\Delta_i}{1-\Delta_i}}\right)^2. \qquad \textbf{[10]}$$

Claim 2. *Assume without loss of generality that $\frac{\mu_1}{\Delta_1} \le \frac{\mu_2}{\Delta_2} \le \dots \le \frac{\mu_n}{\Delta_n}$. Then F_1 has a unique global maximizer $\mathbf{p}^*$ in the simplex $\mathcal{Q} \cap \{\sum_i p_i = 1\}$. Furthermore, there exists $\Lambda \in \{0, \dots, n-1\}$ such that $p_i^* = 0$ if and only if $i \le \Lambda$.*

Proof: Suppose that an optimal solution $\mathbf{p}^*$ has $p_i^* > 0$ whereas $p_j^* = 0$ for some $i < j$. By Eq. **8** we have

$$\tfrac{\partial F_1}{\partial p_i}\big|_{\{p_i=0\}} = \mu_i/\Delta_i, \quad \tfrac{\partial F_1}{\partial p_j}\big|_{\{p_j=0\}} = \mu_j/\Delta_j.$$

Since $\partial F_1/\partial p_i$ is monotone decreasing in p_i we have

$$\tfrac{\partial F_1}{\partial p_i}(\mathbf{p}^*) < \mu_i/\Delta_i \le \mu_j/\Delta_j,$$

where the last inequality used that $i < j$. Hence, shifting a sufficiently small mass $\epsilon > 0$ from p_i^* to p_j^* would increase the value of F_1, contradicting our optimality assumption on $\mathbf{p}^*$.

We have thus established the existence of $\Lambda(\mathbf{p}^*)$ as in the statement. Considering the smallest possible Λ, and recalling F_1 is strictly concave, now implies the uniqueness of $\mathbf{p}^*$. □

The above claim already implies an algorithm—albeit a suboptimal one—for finding the unique feasible optimum of F_1:

- Sort the variables as in *Claim 2*, and for each $\Lambda \in \{0, \dots, n-1\}$, compute λ according to Eq. **10** restricted to the variables $\{p_{\Lambda+1}, \dots, p_n\}$ (setting $p_i = 0$ for $i \le \Lambda$), thus obtaining the values of p_i $(i > \Lambda)$ via Eq. **9**.
- If some Λ yields $0 \le p_i \le 1$ for all $i \in \{\Lambda+1, \dots, n\}$, this is guaranteed to be the unique optimal solution.

Since each iteration over Λ involves $O(n-\Lambda)$ steps, this algorithm has a time complexity of $O(n^2)$.

It is natural to ask how the outputs of the various iterations on Λ relate to the final optimal solution and, namely, whether variables that are infeasible in a given iteration are necessarily 0 in the final optimum. The next theorem establishes that indeed this is the case, providing a faster algorithm for finding the optimum of F_1 subject to the required constraints.

Corollary 3. *Let $\mathbf{p}^* = (p_1^*, \dots, p_n^*)$ be the optimal solution to Problem 1, and assume that $\frac{\mu_1}{\Delta_1} \le \frac{\mu_2}{\Delta_2} \le \dots \le \frac{\mu_n}{\Delta_n}$. Let $\mathbf{p} = (p_1, \dots, p_n)$ be the solution of* Eqs. **9** and **10**. *If $p_1 \le 0$ then necessarily $p_1^* = 0$.*

Proof: This follows immediately from *Claim 2* since, if $p_1^* > 0$, then $\Lambda = 0$ in that claim, thus $p_i = p_i^* > 0$ for all i, as in that case the global maximum is attained in the interior of $\mathcal{Q}$. □

Rearranging Eq. **9**, observe that $p_i \leq 0$ if and only if $\mu_i / \Delta_i \leq \lambda$. In view of the above corollary, if $p_1 \leq 0$ in $\mathbf{p}$ (or equivalently, $\mu_1/\Delta_1 \leq \lambda$) then we may obtain the optimal $\mathbf{p}^*$ by excluding p_1 from the equations and re-solving the system—equivalent to setting $\mu_1 = \Delta_1 = 0$ in Eqs. **9** and **10**. Applying this recursively establishes that *Algorithm 1* indeed solves *Problem 1*.

To solve *Problem 2*, bearing in mind that again F_2 is concave on $\mathcal{Q}$, we can repeat the calculation of the corresponding Lagrange multipliers and obtain the following analogues of Eqs. **9** and **10**:

$$\rho_i = \sqrt{\frac{\mu_i \Delta_i}{\lambda}} - \Delta_i, \qquad \lambda = \left(\frac{\sum_i \sqrt{\mu_i \Delta_i}}{R + \sum_i \Delta_i} \right)^2. \qquad \textbf{[11]}$$

As the arguments of *Claim 2* and *Corollary 3* remain valid for this setting, we deduce that *Algorithm 2* solves *Problem 2* in time $O(n \log n)$. This concludes the proof of *Theorem 1*. □

Remark 4 (nonbinary page freshness): The analysis of the binary freshness extends to a notion of exponentially decreasing freshness, as formulated next, analogous to Eq. **1**. Given a series of requests $(I_1, T_1), \ldots, (I_N, T_N)$, where at time T_j the user requests page I_j (the request rate model), the goal of the server is to maximize

$$\mathbb{E}\left[\frac{1}{N} \sum_{j=1}^{N} \exp\left(-\Delta_i \, A(I_j, T_j)\right) \right], \qquad \textbf{[12]}$$

where the age of page I_j at time T_j is given by

$$A(I_j, T_j) = T_j - \mathrm{LastFetch}_{I_j}(T_j),$$

and $\mathrm{LastFetch}_i(t)$ is the last time page i was fetched before time t (or $-\infty$ if it was never fetched). Since the expression in Eq. **4** (binary changes in continuous time) equals the one in Eq. **12** (exponentially decaying freshness in discrete time), *Theorem 1* implies that *Algorithm 2* finds the unique optimum of the optimization problem corresponding to the latter.

Derandomizing the Optimal Random Policy

Cho and Garcia-Molina (ref. 1, section 4.4) compared the performances of page updates via deterministic, random, and semi-random policies (referred to as fixed-order, random-order, and purely random; in the special case where the update rates are all equal, these correspond to repeatedly cycling through an ordered list of all pages ref. 1, alg. 4.1; repeatedly drawing a random permutation of the pages and going through it ref. 1, alg. 4.2; and repeatedly sampling an i.i.d. uniform page ref. 1, alg. 4.3, respectively), showing the advantage of the deterministic one (indeed, when comparing Eqs. **5** and **6**, one has $\overline{F}(\rho) \geq F_2(\rho)$ for every Δ, ρ, as the inequality $\frac{1-\exp(-x)}{x} \geq \frac{1}{1+x}$ reduces to $1 + x \leq e^x$ which holds for all x). It is thus advantageous to transform the optimal stochastic solution $\rho = (\rho_1, \ldots, \rho_n)$ into a carousel where the frequency of page i is approximately ρ_i. An efficient way to form such a carousel is the earliest-deadline-first (EDF) policy, introduced in ref. 15 in the context of periodic task scheduling. In our setting, this corresponds to the following:

Algorithm 3: EDF Derandomization

```
**Input:** Optimal random policy $(\rho_i)_{i=1}^n$, and $\epsilon > 0$.
**Output:** A sequence $(\omega_1, \ldots, \omega_T)$ for $T = \lceil \epsilon^{-1} \rceil$ such that, if
we let $N_{k,t} := \#\{i \leq t : \omega_i = k\}$, then
  $\max_k |\rho_k - N_{k,T}/T| \leq \epsilon$.
  1. $\delta_k \leftarrow 0$ for $k = 1, \ldots, n$.
  2. for $t = 1$ to $T$ do.
  3. $\delta_k \leftarrow \delta_k + \rho_k$ for $k = 1, \ldots, n$.
  4. $S \leftarrow \{k : \delta_k > 0\}$ and $D_k \leftarrow \lceil (1 - \delta_k)/\rho_k \rceil$ for $k \in S$.
  5. $k_0 \leftarrow argmin\{D_k : k \in S\}$ (breaking ties arbitrarily).
  6. $\omega_t \leftarrow k_0$ and $\delta_{k_0} \leftarrow \delta_{k_0} - 1$.
  7. Return $(\omega_1, \ldots, \omega_T)$.
```

This algorithm is guaranteed to satisfy $\max_{k,t} |t\rho_k - N_{k,t}| \leq 1$ for all $t = 1, \ldots, T$ (see ref. 16, thm. 3, as well as ref. 17) and runs in time $O(n/\epsilon)$. Fig. 1 depicts the performance of the derandomized policy.

1. Cho J, Garcia-Molina H (2003) Effective page refresh policies for web crawlers. *ACM Trans Database Syst* 28:390–426.
2. Cho J, Garcia-Molina H (2000) Synchronizing a database to improve freshness. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000*, Dallas, TX (Association for Computing Machinery, New York), pp. 117–128.
3. Cho J, Garcia-Molina H (2003) Estimating frequency of change. *ACM Trans Internet Technol* 3:256–290.
4. Coffman EG, Liu Z, Weber RR (1998) Optimal robot scheduling for web search engines. *J Scheduling* 1:15–29.
5. Castillo C (2004) Effective web crawling. PhD thesis (University of Chile, Santiago). Available at www.dcc.uchile.cl/tesis/doctorado/Castillo_Ocaranza.pdf.
6. Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web caching and Zipf-like distributions: Evidence and implications. *Proceedings of the 1999 IEEE INFOCOM* (IEEE, New York), pp. 126–134.
7. Bonato A (2008) *A Course on the Web Graph*, Graduate Studies in Mathematics (American Mathematical Society, AARMS, Providence, RI), Vol 89, pp. xii, 184.
8. Horvitz E (1998) Continual computation policies for utility-directed prefetching. *Proceedings of the 1998 ACM Conference on Information and Knowledge Management* (Association for Computing Machinery, New York), pp. 175–184.
9. Horvitz E (2001) Principles and applications of continual computation. *Artif Intell* 126:159–196.
10. Shahaf D, Horvitz E (2009) Investigations of continual computation in IJCAI 2009, *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, Pasadena, CA, USA (Association for the Advancement of Artifical Intelligence, Palo Alto, CA), July 11-17, 2009, pp. 285–291.
11. Gal A, Eckstein J (2001) Managing periodically updated data in relational databases: A stochastic modeling approach. *J ACM* 48:1141–1183.
12. Sia KC, Cho J, Cho H (2007) Efficient monitoring algorithm for fast news alerts. *IEEE Trans Knowl Data Eng*19:950–961.
13. Anily S, Glass CA, Hassin R (1998) The scheduling of maintenance service. *Discrete Appl Math* 82:27–42.
14. Bar-Noy A, Bhatia R, Naor J, Schieber B (2002) Minimizing service and operation costs of periodic scheduling. *Math Oper Res* 27:518–544.
15. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. *J ACM* 20:46–61.
16. Tijdeman R (1980) The chairman assignment problem. *Discrete Math* 32:323–330.
17. Angel O, Holroyd AE, Martin JB, Propp J (2009) Discrete low-discrepancy sequences. arXiv:0910.1077.