Staying up to Date with Online Content Changes Using Reinforcement Learning for Scheduling

Andrey Kolobov
Microsoft Research
Redmond, WA-98052
akolobov@microsoft.com

Yuval Peres
yperes@gmail.com

Cheng Lu
Microsoft Bing
Bellevue, WA-98004
Cheng.Lu@microsoft.com

Eric Horvitz
Microsoft Research
Redmond, WA-98052
horvitz@microsoft.com

Abstract

From traditional Web search engines to virtual assistants and Web accelerators, services that rely on online information need to continually keep track of remote content changes by explicitly requesting content updates from remote sources (e.g., web pages). We propose a novel optimization objective for this setting that has several practically desirable properties, and efficient algorithms for it with optimality guarantees even in the face of mixed content change observability and initially unknown change model parameters. Experiments on 18.5M URLs crawled daily for 14 weeks show significant advantages of this approach over prior art.

1 Introduction

As the Web becomes more and more dynamic, services that rely on web data face the increasingly challenging problem of keeping up with online content changes. Whether it be a continuous-query system [26], a virtual assistant like Cortana or Google Now, or an Internet search engine, such a service tracks many remote sources of information – web pages or data streams [27]. Users expect these services, which we call *trackers*, to surface the latest information from the sources. This is easy if sources *push* content updates to the tracker, but few sources do. Instead, major trackers such as search engines must continually decide when to re-*pull* (*crawl*) data from sources to pick up the changes. A policy that makes these decisions well solves the *freshness crawl scheduling problem*.

Freshness crawl scheduling has several challenging aspects. For most sources, the tracker finds out whether they have changed only when it crawls them. To guess when the changes happen, and hence should be downloaded, the tracker needs a predictive model whose parameters are initially unknown. Thus, the tracker needs to learn these models *and* optimize a freshness-related objective when scheduling crawls. For some web pages, however, sitemap polling and other means can provide trustworthy near-instantaneous signals that the page has changed in *a* meaningful way, though not what the change is exactly. But even with these remote change observations and known change model parameters, freshness crawl scheduling remains highly nontrivial because the tracker cannot react to every individual predicted or actual change. The tracker's infrastructure imposes a *bandwidth constraint* on the average daily number of crawls, usually just a fraction of the change event volume. Last but not least, Google and Bing track many billions of pages [32] with vastly different importance and change frequency characteristics. The sheer size of this constrained learning and optimization problem makes low-polynomial algorithms for it a must, despite the availability of big-data platforms.

This paper presents a holistic approach to freshness crawl scheduling that handles all of the above aspects in a computationally efficient manner with optimality guarantees using a type of reinforcement

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

learning (RL) [29]. This problem has been studied extensively from different angles, as described in the Related Work section. The scheduling aspect per se, under various objectives and assuming known model parameters, has been the focus of many papers, e.g., [2, 10, 13, 25, 35]. In distinction from these works, our approach has *all* of the following properties: (i) optimality; (ii) computational efficiency; (iii) guarantee that every source changing at a non-zero rate will be occasionally crawled; (iv) ability to take advantage of remote change observations, if available. No other work has (iv), and only [35] has (i)-(iii). Moreover, learning change models previously received attention [11] purely as a preprocessing step. Our RL approach integrates it with scheduling, with convergence guarantees.

Specifically, our contributions are: (1) A natural freshness optimization objective based on harmonic numbers, and analysis showing how its mathematical properties enable efficient optimal scheduling. (2) Efficient optimization procedures for this bandwidth-constrained objective under complete, mixed, and lacking remote change observability. (3) A reinforcement learning algorithm that integrates these approaches with model estimation of [11] and converges to the optimal policy, lifting the known-parameter assumption. (4) An approximate crawl scheduling algorithm that requires learning far fewer parameters, and identifying a condition under which its solution is optimal.

2 Problem formalization

In settings we consider, a service we call *tracker* monitors a set W of information sources. A source $w \in W$ can be a web page, a data stream, a file, etc, whose content occasionally changes. To pick up changes from a source, the tracker needs to *crawl* it, i.e., download its content. When source w has changes the tracker hasn't picked up, the tracker is *stale* w.r.t. w; otherwise, it is *fresh* w.r.t. w. We assume near-instantaneous crawl operations, and a fixed set of sources W. Growing W to improve *information completeness* [27] is also an important but distinct problem; we do not consider it here.

Discrete page changes. We define a *content change* at a source as an alteration at least minimally important to the tracker. In practice, trackers compute a source's content *digest* using data extractors, *shingles* [4], or *similarity hashes* [7], and consider content changed when its digest changes.

Models of change process and importance. We model each source $w \in W$'s changes as a Poisson process with *change rate* Δ_w. Many prior works adopted it for web pages [2, 8, 9, 10, 11, 12, 35] as a good balance between fidelity and computational convenience. We also associate an *importance* score μ_w with each source, and denote these parameters jointly as $\vec{\mu}$. Importance score μ_w can be thought of as characterizing the time-homogeneous Poisson rate at which the page is served in response to the query stream, although in general it can be any positive weight measuring source significance [2]. ***While scores μ_w are defined by, and known to, the tracker, change rates Δ_w need to be learned.***

Change observability. For most sources, the tracker can find out whether the source has changed only by crawling it. In this case, even crawling doesn't tell the tracker how many times the source has changed since the last crawl. We denote the set of these sources as W^- and say that the tracker receives *incomplete change observations* about them. However, for other sources, which we denote as W^o, the tracker may receive near-instant notification whenever they change, i.e., get *complete remote change observations*. E.g., for web pages these signals may be available from browser telemetry or sitemaps. Thus the tracker's set of sources can be represented as $W = W^o \cup W^-$ and $W^o \cap W^- = 0$.

Bandwidth constraints. Even if the tracker receives complete change observations, it generally cannot afford to do a crawl upon each of them. The tracker's network infrastructure and considerations of respect to other Internet users limit its *crawl rate* (the average number of requests per day); the total *change rate* of tracked sources may be much higher. We call this limit *bandwidth constraint* R.

Optimizing freshness. The tracker operates in continuous time and starts fresh w.r.t. all sources. Our scheduling problem's solution is a policy π — a rule that at every instant t chooses (potentially stochastically) a source to crawl or decides that none should be crawled. Executing π produces a *crawl sequence* of time-source pairs $CrSeq = (t_1, w_1), (t_2, w_2), \ldots$, denoted $CrSeq_w = (t_1, w), (t_2, w), \ldots$ for a specific source w. Similarly, the (Poisson) change process at the sources generates a change sequence $ChSeq = (t'_1, w'_1), (t'_2, w'_2), \ldots$, where t'_i is a change time of source w'_i; its restriction to source w is $ChSeq_w$. We denote the joint process governing changes at all sources as $P(\vec{\Delta})$.

3 Minimizing harmonic staleness penalty

We view maximizing freshness as minimizing costs the tracker incurs for the lack thereof, and associate the *time-averaged expected staleness penalty* J^π with every scheduling policy π:

$$J^\pi = \lim_{T\to\infty} \mathop{\mathbb{E}}_{\substack{CrSeq\sim\pi,\\ ChSeq\sim P(\vec{\Delta})}} \left[\frac{1}{T}\int_0^T \sum_{w\in W} \mu_w C(N_w(t))dt\right] \tag{1}$$

Here, T is a planning horizon, $N_w(t)$ is the number of uncrawled changes source w has accumulated by time t, and $C : \mathbb{Z}^+ \to \mathbb{R}^+$ is a *penalty function*, to be chosen later, that assigns a cost to every possible number of uncrawled changes. Note that $N_w(t)$ implicitly depends on the most recent time w was crawled as well as on change sequence $ChSeq$, so the expectation is both over possible change sequences *and* possible crawl sequences $CrSeq$ generatable by π. Minimizing staleness means finding $\pi^* = \text{argmin}_{\pi\in\Pi} J^\pi$ under bandwidth constraint, where Π is a suitably chosen policy class.

Choosing $C(n)$ that is efficient to optimize *and* induces "well-behaving" policies is of utmost importance. E.g., $C(n) = \mathbb{1}_{n>0}$, which imposes a fixed penalty if a source has any changes since last crawl [2, 10], can be optimized efficiently in $O(|W|\log(|W|))$ time over the class Π of policies that crawl each source w according to a Poisson process with a source-specific rate ρ_w. However, for many sources, the optimal ρ^*_w is 0 under this $C(n)$ [2]. This is unacceptable in practice, as it leaves the tracker stale w.r.t. some sources forever, raising a question: why monitor these sources at all?

In this paper, we propose and analyze the following penalty:

$$C(n) = H(n) = \sum_{i=1}^{n} \frac{1}{i} \text{ if } n > 0\text{, and } 0 \text{ if } n = 0 \tag{2}$$

$H(n)$ for $n > 0$ is the n-th harmonic number and has several desirable properties as staleness penalty:

It is strictly monotonically increasing. Thus, it penalizes the tracker for every change that happened at a source since the previous crawl, not just the first one as in [10].

It is discrete-concave, providing diminishing penalties: intuitively, while all undownloaded changes at a source matter, the first one matters most, as it marks the transition from freshness to staleness.

"Good" policies w.r.t. this objective don't starve any source as long as that source changes. This is because, as it turns out, policies that ignore changing sources incur $J^\pi = \infty$ if $C(n)$ is as in Eq. 2 (see Prop. 1 in Section 4). In fact, this paper's optimality results and high-level approaches are valid for any concave $C(n) \geq 0$ s.t. $\lim_{n\to\infty} C(n) = \infty$, though possibly at a higher computational cost.

It allows for efficiently finding optimal policies under practical policy classes. Indeed, $C(n) = H(n)$ isn't the only penalty function satisfying the above properties. For instance, $C(n) = n^d$ for $0 < d < 1$ and $C(n) = \log_d(1 + n)$ for $d > 1$ behave similarly, but result in much more computationally expensive optimization problems, as do other alternatives we have considered.

4 Optimization under known change process

We now derive procedures for optimizing Eq. 1 with $C(n) = H(n)$ (Eq. 2) under the bandwidth constraint for sources with incomplete and complete change observations, assuming that we know the change process parameters $\vec{\Delta}$ exactly. In Section 5 we will lift the known-parameters assumption. We assume $\vec{\mu}, \vec{\Delta} > 0$, because sources that are unimportant or never change don't need to be crawled.

4.1 Case of incomplete change observations

When the tracker can find out about changes at a source only by crawling it, we consider randomized policies that sample crawl times for each source w from a Poisson process with rate ρ_w:

$$\Pi^- = \{CrSeq_w \sim Poisson(\rho_w)\ \forall w \in W^- | \vec{\rho} \geq 0\} \tag{3}$$

This policy class reflects the intuition that, since each source changes according to a Poisson process, i.e., roughly periodically, it should also be crawled roughly periodically. In fact, as Azar et al. [2] show, any $\pi \in \Pi^-$ can be de-randomized into a deterministic policy that is approximately periodic for each w. Since every $\pi \in \Pi^-$ is fully determined by the corresponding vector $\vec{\rho}$, we can easily express a bandwidth constraint on $\pi \in \Pi^-$ as $\sum_{w\in W^-} \rho_w = R$.

To optimize over Π^-, we first express policy cost (Eq. 1) in terms of Π^-'s policy parameters $\vec{\rho} \geq 0$:

Proposition 1. *For $\pi \in \Pi^-$, J^π from Eq. 1 is equivalent to*

$$J^\pi = -\sum_{w \in W^-} \mu_w \ln\left(\frac{\rho_w}{\Delta_w + \rho_w}\right) \tag{4}$$

Proof. See the Supplement. Note that $J^\pi = \infty$ if $\rho_w = 0$ for any $w \in W^-$. The proof relies on properties of Poisson processes, particularly memorylessness. ∎

Thus, finding $\pi^* \in \Pi^-$ can be formalized as follows:

Problem 1. *[Finding $\pi^* \in \Pi^-$]*

INPUT: *bandwidth $R > 0$; positive importance and change rate vectors $\vec{\mu}, \vec{\Delta} > 0$.*

OUTPUT: *Crawl rates $\vec{\rho} = (\rho_w)_{w \in W^-}$ maximizing $\overline{J}^\pi = -J^\pi = \sum_{w \in W^-} \mu_w \ln\left(\frac{\rho_w}{\Delta_w + \rho_w}\right)$ subject to $\sum_{w \in W^-} \rho_w = R$, $\rho_w \geq 0$ for all $w \in W^-$.*

The next result readily identifies the optimal solution to this problem:

Proposition 2. *For $\vec{\mu}, \vec{\Delta} > 0$, policy $\pi^* \in \Pi^-$ parameterized by $\vec{\rho}^* > 0$ that satisfies the following equation system is unique, minimizes harmonic penalty J^π in Eq. 2, and is therefore optimal in Π^-:*

$$\begin{cases} \rho_w = \frac{\sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}} - \Delta_w}{2}, \text{for all } w \in W^- \\ \sum_{w \in W^-} \rho_w = R \end{cases} \tag{5}$$

Algorithm 1: LAMBDACRAWL-INCOMLOBS: finding the optimal crawl scheduling policy $\pi^* \in \Pi^-$ under incomplete change observations (Problem 1)

Input: $R \geq 0$ – bandwidth;
$\vec{\mu} > 0, \vec{\Delta} > 0$ – importance and change rates;
$\epsilon > 0$ – desired precision on λ
Output: $\vec{\rho}$ – vector of crawl rates for each source.

$\lambda_{lower} \hookleftarrow \frac{|W^-|^2 \min_{w \in W^-}\{\Delta_w\} \min_{w \in W^-}\{\mu_w\}}{|W^-| \max_{w \in W^-}\{\Delta_w\} R + R^2}$
$\lambda_{upper} \hookleftarrow \frac{|W^-|^2 \max_{w \in W^-}\{\Delta_w\} \max_{w \in W^-}\{\mu_w\}}{|W^-| \min_{w \in W^-}\{\Delta_w\} R + R^2}$
$\lambda \hookleftarrow$ BisectionSearch$(\lambda_{lower}, \lambda_{upper}, \epsilon)$
// see, e.g., Burden & Faires [6]
foreach $w \in W^-$ **do** $\rho_w \hookleftarrow \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2}$
Return $\vec{\rho}$

Proof. See the Supplement. The main insight is that for any $\vec{\mu}, \vec{\Delta} > 0$ the Lagrange multiplier method, which gives rise to Eq. system 5, identifies the only maximizer of $\overline{J}^\pi = -J^\pi$ (Eq. 4) with $\vec{\rho} > 0$, which thus must correspond to $\pi^* \in \Pi^-$. Crucially, that solution always has $\lambda > 0$. ∎

Eq. system 5 is non-linear, but the r.h.s. of Eqs. involving λ monotonically decreases in $\lambda > 0$, so, e.g., bisection search [6] on $\lambda > 0$ can find $\vec{\rho}^*$ as in Algorithm 1.

Proposition 3. LAMBDACRAWL-INCOMLOBS *(Algorithm 1) finds an ϵ-approximation to Problem 1's optimal solution in time $O(\log_2(\frac{\lambda_{upper} - \lambda_{lower}}{\epsilon})|W^-|)$.*

Proof. See the Supplement. The key step is showing that the solution λ is in $[\lambda_{lower}, \lambda_{upper}]$. ∎

Note that a convex problem like this could also be handled using interior-point methods, but the most suitable ones have higher, cubic per-iteration complexity [5].

4.2 Case of complete change observations

If the tracker receives a notification every time a source changes, the policy class Π^- in Eq. 3 is clearly suboptimal, because it ignores these observations. At the same time, crawling every source on every change signal is unviable, because the total change rate of all sources $\sum_{w \in W^o} \Delta_w$ can easily exceed bandwidth R. These extremes suggest a policy class whose members trigger crawls for only a fraction of the observations, dictated by a source-specific probability p_w:

$$\Pi^o = \{\text{for all } w \in W^o\text{, on each observation } o_w \text{ crawl } w \text{ with probability } p_w \mid 0 \leq \vec{p} \leq 1\} \tag{6}$$

As with Π^-, to find $\pi^* \in \Pi^o$ we first express J^π from Eq. 1 in terms of Π^o's policy parameters $\vec{p}$:

Proposition 4. *For $\pi \in \Pi^o$, J^π from Eq. 1 is equivalent to $J^\pi = -\sum_{w \in W^o} \mu_w \ln(p_w)$ if $\vec{p} > 0$ and $J^\pi = \infty$ if $p_w = 0$ for any $w \in W^o$.*

Proof. See the Supplement. The key insight is that under any $\pi \in \Pi^o$, the number of w's uncrawled changes at time t is geometrically distributed with parameter p_w. ∎

Under any $\pi \in \Pi^o$, the crawl rate ρ_w of any source is related to its change rate Δ_w: every time w changes we get an observation and crawl w with probability p_w. Thus, $\rho_w = p_w \Delta_w$. Also, bandwidth $R > \sum_{w \in W^o} \Delta_w$ isn't sensible, because with complete change observations the tracker doesn't benefit from more crawls than there are changes. Thus, we frame finding $\pi^* \in \Pi^o$ as follows:

Problem 2. *[Finding $\pi^* \in \Pi^o$]*

INPUT: *bandwidth R s.t. $0 < R \leq \sum_{w \in W^o}$; importance and change rate vectors $\vec{\mu}, \vec{\Delta} > 0$.*

OUTPUT: *Crawl probabilities $\vec{p} = (p_w)_{w \in W^o}$ subject to $\sum_{w \in W^o} p_w \Delta_w = R$ and $0 \leq p_w \leq 1$ for all $w \in W^o$, maximizing $\overline{J}^\pi = -J^\pi = \sum_{w \in W^o} \mu_w \ln(p_w)$.*

Non-linear optimization under inequality constraints could generally take exponential time in the constraint number. Our main result in this subsection is a *polynomial* optimal algorithm for Problem 2.

First, consider a relaxation of Problem 2 that ignores the inequality constraints:

Proposition 5. *The optimal solution $\vec{\hat{p}}^*$ to the relaxation of Problem 2 that ignores inequality constraints is unique and assigns $\hat{p}^*_w = \frac{R\mu_w}{\Delta_w \sum_{w' \in W^o} \mu_{w'}}$ for all $w \in W^o$.*

Proof. See the Supplement. The proof applies Lagrange multipliers. ∎

Our algorithm LAMBDACRAWL-COMPLOBS (Algorithm 2)'s high-level approach is to iteratively (lines 4-14) solve Problem 2's relaxations as in Prop. 5 (lines 5-6), each time detecting sources that *activate* (either meet or exceed) the $p_w \leq 1$ constraints (line 9). (Note that the relaxed solution never has $\hat{p}^*_w \leq 0$.) Our key insight, which we prove in the Supplement, is that any such source has $p^*_w = 1$. Therefore, we set $p^*_w = 1$ for each of them, adjust the overall bandwidth constraint for the remaining sources to $R_{rem} = R - p^*_w \Delta_w = R - \Delta_w$, and remove w from further consideration (lines 10-12). Eventually, we arrive at a (possibly empty) set of sources for which Prop. 5's solution obeys all constraints under the remaining bandwidth (lines 15-16). Since Prop. 5's solution is optimal in this base case, the overall algorithm is optimal too.

Algorithm 2: LAMBDACRAWL-COMPLOBS: finding the optimal crawl scheduling policy $\pi^* \in \Pi^o$ under complete change observations (Problem 2)

1 LAMBDACRAWL-COMPLOBS:
Input: $\vec{\mu}, \vec{\Delta}$ – importance and change rate vectors
2 R s.t. $0 \leq R \leq \sum_{w \in W} \Delta_w$ – bandwidth;
Output: $\vec{p}^*$ – vector specifying optimal per-page crawl probabilities upon receiving a change observation.
3 $W_{rem} \leftarrow W^o$ // remaining sources to consider
4 **while** $W^o_{rem} \neq \emptyset$ **do**
5 **foreach** $w \in W^o_{rem}$ **do**
6 $\hat{p}^*_w \leftarrow \frac{R\mu_w}{\Delta_w \sum_{w' \in W^o_{rem}} \mu_{w'}}$ for all $w \in W^o_{rem}$
7 $ViolationDetected \leftarrow False$
8 **foreach** $w \in W^o_{rem}$ **do**
9 **if** $\hat{p}^*_w \geq 1$ **then**
10 $p^*_w \leftarrow 1$
11 $R \leftarrow R - \Delta_w$ // reduce remaining bandwidth
12 $W^o_{rem} \leftarrow W^o_{rem} \setminus \{w\}$ // ignore w onwards
13 $ViolationDetected = True$
14 **if** $ViolationDetected == False$ **then** break
15 **foreach** $w \in W^o_{rem}$ **do**
16 $p^*_w \leftarrow \hat{p}^*_w$
17 Return $\vec{p}^* = (p^*_w)_{w \in W^o}$

Proposition 6. LAMBDACRAWL-COMPLOBS *is optimal for Problem 2 and runs in time $O(|W^o|^2)$.*

Proof. See the Supplement. The proof critically relies on the concavity of $\overline{J}^\pi$. ∎

The $O(|W^o|^2)$ bound is loose. Each iteration usually discovers several active constraints at once, and for many sources the constraint is never activated, so the actual running time is close to $O(|W^o|)$.

4.3 Crawl scheduling under mixed observability

In practice, trackers have to simultaneously handle sources with and without complete change data under a *common* bandwidth budget R. Consider a policy class that combines Π^- and Π^o:

$$\Pi^\ominus = \begin{cases} \text{For all } w \in W^-\text{:}\{CrSeq_w \sim Poisson(\rho_w) | \vec{\rho}\}, \\ \text{For all } w \in W^o\text{:}\{\text{on each change observation } o_w\text{, crawl } w \text{ with probability } p_w | \vec{p}\} \end{cases} \quad (7)$$

For $\pi \in \Pi^\ominus$, Prop.s 1 and 4 imply that J^π from Eq. 1 is equivalent to

$$J^\pi = -\sum_{w \in W^-} \mu_w \ln\left(\frac{\rho_w}{\Delta_w + \rho_w}\right) - \sum_{w \in W^o} \mu_w \ln(p_w) \quad (8)$$

Optimization over $\pi \in \Pi^{\ominus}$ can be stated as follows:

Problem 3. *[Finding $\pi^* \in \Pi^{\ominus}$]*

INPUT: *bandwidth $R > 0$; importance and change rate vectors $\vec{\mu}, \vec{\Delta} > 0$.*

OUTPUT: *Crawl rates $\vec{\rho} = (\rho_w)_{w \in W^-}$ and crawl probabilities $\vec{p} = (p_w)_{w \in W^o}$ maximizing*

$$\overline{J}^{\pi} = -J^{\pi} = \sum_{w \in W^-} \mu_w \ln\left(\frac{\rho_w}{\Delta_w + \rho_w}\right) + \sum_{w \in W^o} \mu_w \ln(p_w) \tag{9}$$

subj. to $\sum_{w \in W^-} \rho_w + \sum_{w \in W^o} p_w \Delta_w = R, \rho_w > 0$ *for all* $w \in W^-, 0 < p_w \leq 1$ *for all* $w \in W^o$.

The optimization objective (Eq. 9) is strictly concave as a sum of concave functions over the constrained region, and therefore has a unique maximizer. Finding it amounts to deciding how to split the total bandwidth R into R^o for sources with complete change observations and $R^- = R - R^o$ for the rest. For any candidate split, LAMBDACRAWL-COMPLOBS and LAMBDACRAWL-INCOMLOBS give us the reward-maximizing policy parameters $\vec{p}^*(R^o)$ and $\vec{\rho}^*(R^-)$, respectively, and Eq. 9 then tells us the overall value $\overline{J}^*(R^o, R^-)$ of that split. We also know that for the optimal split, $R^{o^*} \in [0, \min\{R, \sum_{w \in W_o} \Delta_w\}]$, as discussed immediately before Problem 2. Thus, we can find Problem 3's maximizer to any desired precision using a method such as Golden-section search [20] on R^o. LAMBDACRAWL (Algorithm 3) implements this idea, where SPLIT-EVAL-$\overline{J}^*$ (line 7) evaluates $\overline{J}^*(R^o, R^-)$ and OptMaxSearch denotes an optimal search method.

Algorithm 3: LAMBDACRAWL: finding optimal mixed-observability policy $\pi^* \in \Pi^{\ominus}$ (Problem 3)

Input: $R > 0$ – bandwidth;
$\vec{\mu} > 0, \vec{\Delta} > 0$ – importance and change rates;
$\epsilon^{\text{no-obs}}, \epsilon > 0$ – desired precisions

Output: $\vec{\rho}^*, \vec{p}^*$ – crawl rates and probabilities for sources without and with complete change observations.

1 $R^o_{min} \leftarrow 0$
2 $R^o_{max} \leftarrow \min\{R, \sum_{w \in W_o} \Delta_w\}$
3 $\vec{\rho}^*, \vec{p}^* \leftarrow$ *OptMaxSearch*(Split-Eval-$\overline{J}^*$, $R^o_{min}, R^o_{max}, \epsilon$)
4 // E.g., Golden section search [20]
5 Return $\vec{\rho}^*, \vec{p}^*$
6
7 SPLIT-EVAL-$\overline{J}^*$:

Input: R^o – bandwidth for sources with complete change observations, $R, \vec{\mu}, \vec{\Delta}, \epsilon^{\text{no-obs}}$

Output: $\overline{J}^*$ (Eq. 9) for the given split

8 $\vec{\rho} \leftarrow$ LAMBDACRAWL-INCOMLOBS($R - R^o, \vec{\mu}_{W^-}, \vec{\Delta}_{W^-}, \epsilon^{\text{no-obs}}$)
9 $\vec{p} \leftarrow$ LAMBDACRAWL-COMPLOBS($R^o, \vec{\mu}_{W^o}, \vec{\Delta}_{W^o}$)
10 Return $\sum_{w \in W^-} \mu_w \ln\left(\frac{\rho_w}{\Delta_w + \rho_w}\right) + \sum_{w \in W^o} \mu_w \ln(p_w)$

Proposition 7. LAMBDACRAWL *(Algorithm 3) finds an ϵ-approximation to Problem 3's optimal solution using $O(\log(\frac{R}{\epsilon}))$ calls to* LAMBDACRAWL-INCOMLOBS *and* LAMBDACRAWL-COMPLOBS.

Proof. This follows directly from the optimality of LAMBDACRAWL-INCOMLOBS and LAMBDACRAWL-COMPLOBS (Prop.s 2 and 6), as well as of OptMaxSearch such as Golden section, which makes $O(\log(\frac{R}{\epsilon}))$ iterations. ■

5 Reinforcement learning for scheduling

All our algorithms so far assume known change rates, but in reality change rates are usually unavailable and vary with time, requiring constant re-learning. In this section we modify LAMBDACRAWL into a model-based reinforcement learning (RL) algorithm that learns change rates on the fly.

For a source w, suppose the tracker observes binary change indicators $\{z_j\}_{j=1}^U$, where $\{t_j\}_{j=0}^U$ are observation times and $z_j = 1$ iff w changed since t_{j-1} *at least* once. Consider two cases:

Incomplete change observations for w. Here, the tracker generates the sequence $\{z_j\}_{j=1}^U$ for each source w by crawling it. If $z_j = 1$, the tracker still doesn't know exactly how many times the source changed since time t_{j-1}. Denoting $a_{t_j} = t_j - t_{j-1}, j \geq 1$, $\hat{\Delta}$ that solves

$$\sum_{j:z_j=1} \frac{a_j}{e^{a_j \Delta} - 1} - \sum_{j:z_j=0} a_j = 0, \tag{10}$$

is an MLE of Δ for the given source [11]. The l.h.s. of the equation is monotonically decreasing in Δ, so $\hat{\Delta}$ can be efficiently found numerically. This estimator is consistent under mild conditions [11], e.g., if the sequence $\{a_j\}_{j=1}^{\infty}$ doesn't converge to 0, i.e., if the observations are spaced apart.

Complete change observations for w. In this case, for all j, $z_j = 1$: an observation indicating *exactly one* change arrives on every change. Here a consistent MLE of Δ is the observation rate [30]:

$$\hat{\Delta} = (U+1)/t_U, \qquad (11)$$

LAMBDALEARNANDCRAWL, a model-based RL version of LAMBDACRAWL that uses these estimators to learn model parameters simultaneously with scheduling is presented in Algorithm 4. It operates in epochs of length T_{epoch} time units each (lines 3-13). At the start of each epoch n, it calls LAMBDACRAWL (Algorithm 3) on the available $\vec{\hat{\Delta}}_{n-1}$ change rate estimates to produce a policy $(\vec{\rho}^*_n, \vec{p}^*_n)$ optimal with respect to them (line 4). Executing this policy during the current epoch, for the time period of T_{epoch}, and recording the observations extends the observation history (lines 7-8). (Note though that for sources $w \in W^o$, the observations don't depend on the policy.) It then re-estimates change rates using a suffix of the augmented observation history (lines 10-13). Under mild assumptions, LAMBDALEARNANDCRAWL converges to the optimal policy:

Algorithm 4: LAMBDALEARNANDCRAWL: finding optimal crawl scheduling policy $\pi^* \in \Pi^{\ominus}$ (Problem 3) under initially unknown change model

---

Input: $R > 0$ – bandwidth;
$\vec{\mu} > 0, \vec{\hat{\Delta}}_0 > 0$ – importance and initial change rate guesses
$\epsilon^{\text{no-obs}}, \epsilon > 0$ – desired precisions
$T_{epoch} > 0$ – duration of an epoch
$N_{epochs} > 0$ – number of epochs
$S(n)$ – for each epoch n, observation history suffix length for learning $\vec{\Delta}$ in that epoch

1 // $obs\_hist[S(n)]$ is $S(n)$-length observation history suffix
2 $obs\_hist \hookleftarrow ()$
3 **foreach** $1 \leq n \leq N_{epochs}$ **do**
4 $\quad \vec{\rho}^*_n, \vec{p}^*_n \hookleftarrow$ LAMBDACRAWL$(R, \vec{\mu}, \vec{\hat{\Delta}}_{n-1}, \epsilon^{\text{no-obs}}, \epsilon)$
5 $\quad$ // $\vec{Z}_{new}$ holds observations for all sources from start to
6 $\quad$ // end of epoch n. Execute policy $(\vec{\rho}^*_n, \vec{p}^*_n)$ to get it
7 $\quad \vec{Z}_{new} \hookleftarrow ExecuteAndObserve(\vec{\rho}^*_n, \vec{p}^*_n, T_{epoch})$
8 $\quad Append(obs\_hist, \vec{Z}_{new})$
9 $\quad$ // Learn new $\vec{\Delta}$ estimates using Eqs. 10 and 11
10 $\quad$ **foreach** $w \in W^-$ **do**
11 $\quad\quad \hat{\Delta}_{n_w} \hookleftarrow Solve\Big(\sum_{j: z_{j_w}=1} \frac{a_j}{e^{a_j \Delta} - 1} + \frac{0.5}{e^{0.5\Delta} - 1} - \sum_{j: z_{j_w}=0} a_j - 0.5 = 0, obs\_hist[S(n)]\Big)$
12 $\quad$ **foreach** $w \in W^o$ **do**
13 $\quad\quad \hat{\Delta}_{n_w} \hookleftarrow Solve\Big(\frac{U_{S(n)}+0.5}{S(n)+0.5}, obs\_hist[S(n)]\Big)$

Proposition 8. LAMBDALEARNANDCRAWL *(Algorithm 4) converges in probability to the optimal policy under the true change rates $\vec{\Delta}$, i.e., $\lim_{N_{epochs}\to\infty}(\vec{\rho}_{N_{epochs}}, \vec{p}_{N_{epochs}}) = (\vec{\rho}^*, \vec{p}^*)$, if $\vec{\Delta}$ is stationary and $S(n)$, the length of the history's training suffix, satisfies $S(N_{epoch}) = length(obs\_hist)$.*

Proof. See the Supplement. It follows from the consistency and positivity of the change rate estimates, as well as LAMBDACRAWL's optimality ■

LAMBDALEARNANDCRAWL in practice requires attention to several aspects:

Stationarity of $\vec{\Delta}$. Source change rates may vary with time, so the length of history suffix for estimating $\vec{\Delta}$ should be shorter than the entire available history.

Singularities of $\vec{\hat{\Delta}}$ estimators. The MLE in Eq. 10 yields $\hat{\Delta}_w = \infty$ if all crawls detect a change (the r.h.s. is 0). Similarly, Eq. 11 produces $\hat{\Delta}_w = 0$ if no observations about w arrive in a given period. To avoid these singularities without affecting consistency, we smooth the estimates by adding imaginary observation intervals of length 0.5 to Eq. 10 and imaginary 0.5 observation to Eq. 11 (lines 11,13).

Number of parameters. Learning a change rate separately for each source can be slow. Instead, we can generalize change rates across sources (e.g., [12]). Alternatively, sometimes we can avoid learning for most pages altogether:

Proposition 9. *Suppose the tracker's set of sources W^- is such that for some constant $c > 0$, $\frac{\mu_w}{\Delta_w} = c$ for all $w \in W^-$. Then minimizing harmonic penalty under incomplete change observations (Problem 1) has $\rho^*_w = \frac{\mu_w R^-}{\sum_{w' \in W^-} \mu_{w'}}$.*

Proof. See the Supplement. The proof proceeds by plugging in $\Delta_w = \frac{1}{c}\mu_w$ into Eq. system 5. ■

Thus, if the importance-to-change-rate ratio is roughly equal across all sources, then their crawl rates don't depend on change rates or even the ratio constant itself. Thus, we don't need to learn them for sources $w \in W^-$ and can hope for faster convergence, although for some quality loss (see Section 7).

6 Related work

Scheduling for Posting, Polling, and Maintenance. Besides monitoring information sources, mathematically related settings arise in smart broadcasting in social networks [19, 31, 33, 36], personalized teaching [31], database synchronization [14], and job and maintenance service scheduling [1, 3, 15, 16]. In web crawling context (see Olston & Najork [22] for a survey), the closest works are [10], [35], [25], and [2]. Like [10] and [2], we use Lagrange multipliers for optimization, and adopt the Poisson change model of [10] and many works since. Our contributions differ from prior art in several ways: (1) optimization objectives (see below) and guarantees; (2) special crawl scheduling under complete change observations; (3) reinforcement learning of model parameters during crawling.

Optimization objectives. Our objective falls in the class of convex separable resource allocation problems [17]. So do most other related objectives: binary freshness/staleness [2, 10], age [8], and embarrassment [35]. The latter is implemented via specially constructed importance scores [35], so our algorithms can be used for it too. Other separable objectives include information longevity [23]. In contrast, Pandey & Olston [25] focus on an objective that depends on user behavior and cannot be separated into contributions from individual sources. While intuitively appealing, their measure can be optimized only via many approximations [25], and the algorithm for it is ultimately heuristic.

Acquiring model parameters. Importance can be defined and quickly determined from information readily available to search engines, e.g., page relevance to queries [35], query-independent popularity such as PageRank [24], and other features [25, 28]. Learning change rates is more delicate. Change rate estimators we use are due to [11]; our contribution in this regard is integrating them into crawl scheduling while providing theoretical guarantees, as well as identifying conditions when estimation can be side-stepped using an approximation (Prop. 9). While many works adopted the homogeneous Poisson change process [2, 8, 9, 10, 11, 12, 35], its non-homogeneous variant [14], quasi-deterministic [35], and general marked temporal point process [31] change models were also considered. Change models can also be inferred via generalization using source co-location [12] or similarity [28].

RL. Our setting could be viewed as a *restless* multi-armed bandit (MAB) [34], a MAB type that allows an arm to change its reward/cost distribution without being pulled. However, no known restless MAB class allows arms to incur a cost/reward without being pulled, as in our setting. This distinction makes existing MAB analysis such as [18] inapplicable to our model. RL with events and policies obeying general marked temporal point processes was studied in [31]. However, it relies on DNNs and as a result doesn't provide guarantees of convergence, optimality, other policy properties, or a mechanism for imposing strict constraints on bandwidth, and is far more expensive computationally.

7 Empirical evaluation

Our experimental evaluation assesses the relative performance of LAMBDACRAWL, LAMBDACRAWLAPPROX, and existing alternatives, evaluates the benefit of using completes change observations, and shows empirical convergence properties of RL for crawl scheduling (Sec. 5). **Please refer to the Supplement, Sec. 9, for details of the experiment setup. All the data and code we used are available at https://github.com/microsoft/Optimal-Freshness-Crawl-Scheduling**.

Metrics. We assessed the algorithms in terms of two criteria. One is the *harmonic policy cost* J_h^π, defined as in Eq. 1 with $C(n)$ as in Eq. 2, which LAMBDACRAWL optimizes directly. The other is the *binary policy cost* J_b^π, also defined as in Eq. 1 but with $C(n) = \mathbb{1}_{n>0}$. It was used widely in previous works, e.g., [2, 10, 35], and is optimized directly by BinaryLambdaCrawl [2]. LAMBDACRAWL doesn't claim optimality for it, but we can still use it to evaluate LAMBDACRAWL's policy.

Data and baselines. The experiments used web page change and importance data collected by crawling $18,532,314$ URLs daily for 14 weeks. We compared LAMBDACRAWL (labeled *LC* in the figures), LAMBDACRAWLAPPROX (*LCA*, *LC* with Prop. 9's approximation), and their RL variants *LLC* (Alg. 4) and *LLCA* to *BinaryLambdaCrawl* (*BLC*) [2], the state-of-the-art optimal algorithm for the binary cost J_b^π. Since *BLC* may crawl-starve sources and hence get $J_h^\pi = \infty$ (see Fig. 1), we also used our own variant, *BLC*ϵ, with the non-starvation guarantee, and its RL flavor *BLLC*ϵ. Finally, we used *ChangeRateCrawl* (*CC*) [10, 35] and *UniformCrawl* (*UC*) [9, 23] heuristics. In each run of an experiment, the bandwidth R was 20% of the total number of URLs used in that run.

Results. We conducted three experiments, whose results support the following claims:

(1) LAMBDACRAWL*'s harmonic staleness cost J_h^π is a more robust objective than the binary cost J_b^π widely studied previously: optimizing the former yields policies that are also near-optimal w.r.t. the latter, while the converse is not true.* In this experiment, whose results are shown in Fig. 1, we

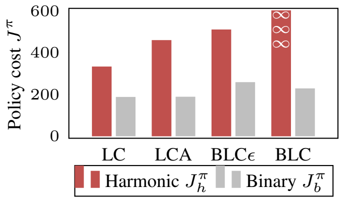

Figure 1: Performance w.r.t. harmonic (J_h^π) and binary (J_b^π) policy costs. **Lower bars = better policies.** *LC* is robust to both, but *BLCϵ* & *BLC* [2] aren't: *LC* (J_h^π-optimal) beats *BLC* ($J_h^\pi = \infty$) and *BLCϵ* by 35% w.r.t. J_h^π, but *BLC* (J_b^π-optimal for incomplete-change-observation URLs) and *BLCϵ* don't beat *LC*/*LCA* w.r.t. J_b^π. *CC* ($J_h^\pi = 2144, J_b^\pi = 963$) and *UC* ($J_h^\pi = 1268, J_b^\pi = 628$) did poorly and were omitted from the plot.

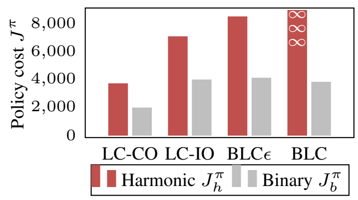

Figure 2: Benefit of using complete change observations. Here we use only the URLs that provide them (4% of our dataset), via sitemaps and other signals. On this URL subset, LAMBDACRAWL reduces to LC-ComplObs (*LC-CO*, Alg. 2) that heeds these signals, while LC-IncomplObs (*LC-IO*, Alg. 1), *BLC*, and *BLCϵ* ignore them. As a result, *LC-CO*'s policy cost both w.r.t. J_h^π and J_b^π is at least 50% lower (!) than the other algorithms'.

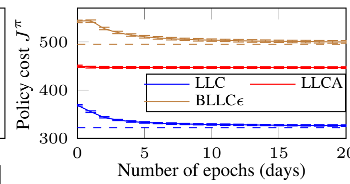

Figure 3: Convergence of the RL-based *LLC*, *LLCA*, and *BLLCϵ* initialized with uniform change rate estimates of 1/day. Dashed lines show asymptotic policy costs (J_h^π of *LC*, *LCA*, and *BLCϵ* from Fig. 1); plots have confidence intervals. *LLC* converges notably faster than *BLLCϵ*, *LLCA* even more so, as it learns fewer parameters. *LLCA*'s asymptotic policy is worse than *LLC*'s but better than *BLLCϵ*'s, especially w.r.t. binary cost J_b^π (Fig. 1).

assumed known change rates. To obtain them, we applied the change rate estimators in Eqs. 10 and 11 to all of the 14-week crawl data for 18.5M URLs, and used their output as ground truth. Policies were evaluated using equations in Props. 1, 4 to get J_h^π, and Eqs. 12, 13 in the Supplement to get J_b^π.

(2) Utilizing complete change observations as LAMBDACRAWL *does when they are available makes a very big difference in policy cost.* Per Fig. 1, *LC* outperforms *BLC* even in terms of binary cost J_b^π, w.r.t. which *BLC* gives an optimatlity guarantee *as long as all URLs have only incomplete change observations*. This raises the question: can *LC*'s and *LCA*'s specialized handling of the complete-observation URLs, a mere 4% of our dataset, explain their overall performance advantage?

The experiment results in Fig. 2 suggest that this is the case. Here we used only the aforementioned URLs with complete change observations. On this URL set, *LC* reduces to *LC-CO* (Alg. 2) and yields a 2× reduction in harmonic cost J_h^π compared to treating these URLs conventionally as *LC-IO* (Alg. 1), *BLC*, and *BLCϵ* do. On the full 18.5M set of URLs, *LC* crawls its complete-observability subset even more effectively by allocating to it a disproportionately large fraction of the overall bandwidth.

Although its handling of complete-observation URLs gives *LC* an edge over alternatives, note that in the hypothetical situation where *LC* treats these URLs conventionally, as reflected in the *LC-IO*'s plot in Fig. 2, it is still at par with *BLC* and *BLCϵ* w.r.t. J_b^π, and markedly outperforms them w.r.t. J_h^π.

(3) When source change rates are initially unknown, the approximate LLCA *converges faster w.r.t. J_h^π than the optimal* LLC*, but at the cost of higher asymptotic policy cost.* Interestingly, *LLCA*'s approximation (Prop. 9) only weakly affects its asymptotic performance w.r.t. binary cost J_b^π (Fig. 1). These factors and algorithm simplicity make this approximation a useful tradeoff in practice.

This experiment, whose analysis is presented in Fig. 3, compared *LLC*, *LLCA*, and *BLLCϵ* in settings where URL change rates have to be learned on the fly. We chose 20 100,000-URL subsamples of our 18.5M-URL dataset randomly with replacement, and used them to simulate 20 21-day runs of each algorithm starting with change rate estimates of 1 change/day for each URL. We used "ground truth" change rates to generate change times for each URL. Every simulated day (epoch; see Alg. 4), each algorithm re-estimated change rates from observations, which were sampled according to the algorithm's current policy, of simulated URL changes. For the next day, it reoptimized its policy for the new rate estimates, and this policy was evaluated with equations in Props. 1, 4 *under the ground-truth rates*. Each algorithm's policy costs across 20 episodes were averaged for each day.

8 Conclusion

We have introduced a new optimization objective and a suite of efficient algorithms for it to address the freshness crawl scheduling problem faced by services from search engines to databases. In particular, we have presented LAMBDALEARNANDCRAWL, which integrates model parameter learning with scheduling optimization. To provide theoretical convergence rate analysis in the future, we intend to frame this problem as a restless multi-armed bandit setting [18, 34].

Acknowledgements. We would like to thank Lin Xiao (Microsoft Research) and Junaid Ahmed (Microsoft Bing) for their comments and suggestions regarding this work.

References

[1] Anily, S., Glass, C., and Hassin, R. The scheduling of maintenance service. *Discrete Applied Mathematics*, pp. 27–42, 1998.

[2] Azar, Y., Horvitz, E., Lubetzky, E., Peres, Y., and Shahaf, D. Tractable near-optimal policies for crawling. *Proceedings of the National Academy of Sciences (PNAS)*, 2018.

[3] Bar-Noy, A., Bhatia, R., Naor, J., and Schieber, B. Minimizing service and operation costs of periodic scheduling. In *SODA*, pp. 11–20, 1998.

[4] Broder, A., Glassman, S., Manasse, M., and Zweig, G. Syntactic clustering of the web. In *WWW*, pp. 1157–1166, 1997.

[5] Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[6] Burden, R. L. and Faires, J. D. *Numerical Analysis*. PWS Publishers, 3rd edition, 1985.

[7] Charikar, M. Similarity estimation techniques from rounding algorithms. In *STOC*, pp. 380–388, 2002.

[8] Cho, J. and Garcia-Molina, H. Synchronizing a database to improve freshness. In *ACM SIGMOD International Conference on Management of Data*, 2000.

[9] Cho, J. and Garcia-Molina, H. The evolution of the web and implications for an incremental crawler. In *VLDB*, 2000.

[10] Cho, J. and Garcia-Molina, H. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, 2003.

[11] Cho, J. and Garcia-Molina, H. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3):256–290, 2003.

[12] Cho, J. and Ntoulas, A. Effective change detection using sampling. In *VLDB*, 2002.

[13] Coffman, E. G., Liu, Z., and Weber, R. R. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1), 1998.

[14] Gal, A. and Eckstein, J. Managing periodically updated data in relational databases. *Journal of ACM*, 48:1141–1183, 2001.

[15] Glazebrook, K. D. and Mitchell, H. M. An index policy for a stochastic scheduling model with improving/deteriorating jobs. *Naval Research Logistics*, 49:706–721, 2002.

[16] Glazebrook, K. D., Mitchell, H. M., and Ansell, P. S. Index policies for the maintenance of a collection of machines by a set of repairmen. *European Journal of Operations Research*, 165 (1):267–284, 2005.

[17] Ibaraki, T. and Katoh, N. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, 1988.

[18] Immorlica, N. and Kleinberg, R. Recharging bandits. In *FOCS*, 2018.

[19] Karimi, M. R., Tavakoli, E., Farajtabar, M., Song, L., and Gomez-Rodriguez, M. Smart broadcasting: Do you want to be seen? In *ACM KDD*, 2016.

[20] Kiefer, J. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.

[21] Kolobov, A., Peres, Y., Lubetzky, E., and Horvitz, E. Optimal freshness crawl under politeness constraints. In *SIGIR*, 2019.

[22] Olston, C. and Najork, M. Web crawling. *Foundations and Trends in Information Retrieval*, 3 (1):175–246, 2010.

[23] Olston, C. and Pandey, S. Recrawl scheduling based on information longevity. In *WWW*, pp. 437–446, 2008.

[24] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, MA, USA, 1998.

[25] Pandey, S. and Olston, C. User-centric web crawling. In *WWW*, 2005.

[26] Pandey, S., Ramamritham, K., and Chakrabarti, S. Monitoring the dynamic web to respond to continuous queries. In *WWW*, 2003.

[27] Pandey, S., Dhamdhere, K., and Olston, C. WIC: A general-purpose algorithm for monitoring web information sources. In *VLDB*, 2004.

[28] Radinsky, K. and Bennett, P. N. Predicting content change on the web. In *WSDM*, pp. 415–424, 2013.

[29] Sutton, R. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

[30] Taylor, H. and Karlin, S. *An Introduction To Stochastic Modeling*. Academic Press, 3rd edition, 1998.

[31] Upadhyay, U., De, A., and Gomez-Rodriguez, M. Deep reinforcement learning of marked temporal point processes. In *NeurIPS*, 2018.

[32] van den Bosch, A., Bogers, T., and de Kunder, M. A longitudinal analysis of search engine index size. In *ISSI*, 2015.

[33] Wang, Y., Williams, G., and Theodorou, E. Variational policy for guiding point processes. In *ICML*, 2017.

[34] Whittle, P. Restless bandits: Activity allocation in a changing world. *Applied Probability*, 25 (A):287–298, 1988.

[35] Wolf, J. L., Squillante, M. S., Yu, P. S., Sethuraman, J., and Ozsen, L. Optimal crawling strategies for web search engines. In *WWW*, 2002.

[36] Zarezade, A., Upadhyay, U., Rabiee, H. R., and Gomez-Rodriguez, M. Redqueen: An online algorithm for smart broadcasting in social networks. In *ACM KDD*, 2017.

SUPPLEMENT

9 Details of the Experiments and Additional Plots

9.1 Dataset, Implementation, Hardware

For the dataset, we crawled $18,532,314$ URLs daily over 14 weeks to estimate their change rates reliably using Equations 10 and 11. Some of the URL crawls on some days failed for reasons ranging from crawler's internal errors to the URL host being temporarily unavailable, so many URLs were crawled fewer than $14 \cdot 7 = 98$ times. At the same time, some URLs were crawled more often as part of the crawler's other workloads.

These URLs are data sources for the knowledge base of a major virtual assistant. The knowledge base uses special information extractors to get important information out of these pages. To determine if a page changed, we ran the same information extractors on it every time we crawled it and considered the page as changed if the extracted information changed.

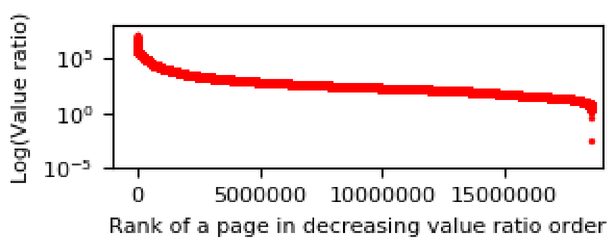

Figure 4: Ranking of $18,532,314$ URLs in our dataset by their $\frac{\mu_w}{\Delta_w}$ ratio.

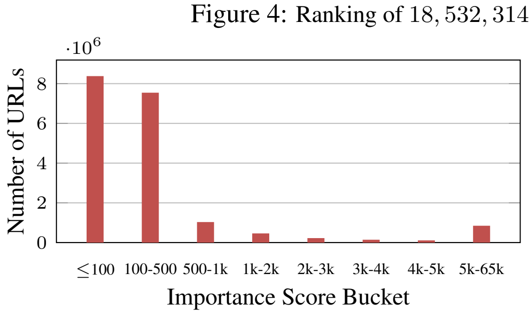

Figure 5: Importance score histogram for URLs in our dataset. The distribution has a big skew, with most pages having importance less than 1000.

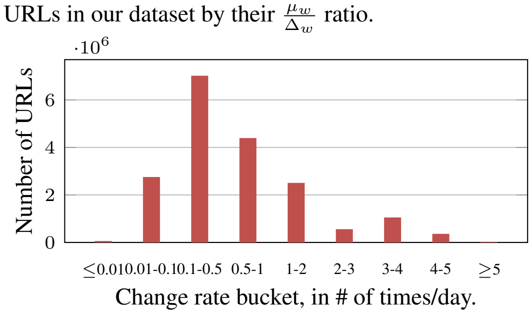

Figure 6: (Poisson) change rate distribution for URLs in our dataset. Most URLs change once in a few days.

4% of the URLs in the dataset had near-complete change observations that we obtained by frequently crawling reliable sitemaps associated with these URLs and using other signals. The near-completeness is due to the fact that we considered a web page changed only if certain information extracted from it changed, which is usually different from a sitemap maintainer's notion of change. As a result, a sitemap could report a change that we wouldn't consider as one and, conversely, could fail to report changes important to us.

We set the URL importance scores μ_w to values defined by the production crawler based on PageRank and popularity.

Proposition 9 suggests that the performance gap between LAMBDACRAWL and LAMBDACRAWLAPPROX depends on the distribution of the ratios $\frac{\mu_w}{\Delta_w}$ across the set of sources W: if they are all equal, the policy cost of LAMBDACRAWL and LAMBDACRAWLAPPROX should be the same. As Figure 4 shows, these ratios are similar across much of our dataset, but clearly non-uniform in the head and tail of the ranking. Figures 5 and 6 show the dataset's importance and change rate distributions.

In each run of an algorithm, **the bandwidth constraint was set to 20% of the number of pages used in that run.**

All algorithms used in the experiments were implemented in Python and run on a Windows 10 laptop with 16GB RAM and an Intel quad-core 2.11GHz i7-8650U CPU. The implementations are available at https://github.com/microsoft/Optimal-Freshness-Crawl-Scheduling.

9.2 Evaluation metrics

To evaluate the algorithms' performance, we used two metrics:

- **The harmonic policy cost** J_h^π as in Equation 1 with $C(n)$ as in Equation 2.
- **The binary policy cost** J_b^π as in Equation 1 with $C(n)$ defined as

$$C(n) = \mathbb{1}_{n>0}$$

.

This policy cost objective was studied in several works including [2, 10]. Some prior research considered its finite-horizon [35] and discrete-time versions [2]. Note that some of these papers formulated their objective as *maximizing freshness*, whereby the agent is *rewarded* for each time unit when the number of accumulated changes at a source is 0. Maximizing this objective means minimizing binary staleness (although the two aren't necessarily negations of each other!) Thus, the two are equivalent and we don't distinguish between them in the paper.

Actually using J_b^π to evaluate policy requires deriving its parameterization in terms of policy π's crawl rates $\vec{\rho}$ and crawl probabilities $\vec{\pi}$, analogously to Propositions 1 and 4 for the harmonic cost J_h^π. By following the steps in the proofs of these propositions, we derived the following expressions for J_b^π:

$$J_b^\pi = \sum_{w \in W^-} \frac{\mu_w \Delta_w}{\Delta_w + \rho_w} \quad \text{for pages } w \in W^- \tag{12}$$

$$J_b^\pi = \sum_{w \in W^o} \mu_w (1 - p_w) \quad \text{for pages } w \in W^o \tag{13}$$

Note that Equation 12 is similar to the equation in Azar et al. [2] for evaluating the *freshness reward* of policy π.

For each experiment, we report the values of J_h^π and J_b^π normalized by the number of URLs used in that experiment.

9.3 Algorithms

In the experiments description in Section 7, we refer to the following algorithms used in the empirical evaluation:

LAMBDACRAWL (*LC*), as in Algorithm 3.

LAMBDACRAWL-COMPLOBS (LC-ComplObs, *LC-CO*), as in Algorithm 2.

LAMBDACRAWL-INCOMLOBS (LC-IncomplObs, *LC-IO*), as in Algorithm 1.

***BinaryLambdaCrawl* (*BLC*)** is the name we give to the state-of-the-art, optimal algorithm proposed by Azar et al. [2] for minimizing binary staleness J_b^π. *BLC* is one of two major baselines for LAMBDACRAWL in our experiments.

The difference between *BLC*'s objective J_b^π and *LC*'s objective J_h^π is crucial in practice, because minimizing binary staleness J_b^π generally yields $\vec{\rho}^*$ with $\rho_w^* = 0$ for many sources even if they have $\Delta_w > 0$. This effectively tells the tracker to ignore changes to these sources — an unacceptable strategy in real applications. Indeed, harmonic penalty (Equation 1) assigns $J_h^\pi = \infty$ to such strategies.

***BinaryLambdaCrawl*(ϵ) (*BLC*ϵ)** Vanilla *BLC*'s lack of non-starvation guarantees makes comparing it to LAMBDACRAWL in terms of J_h^π uninsightful, because for *BLC*, J_h^π is usually ∞. To address this issue, and simultaneously make *BLC* more practical, we modified *BLC* to enforce the non-starvation guarantee. The resulting algorithm is *BLC*ϵ.

Namely, *BLC*ϵ accepts a parameter $\epsilon \in [0, 1]$. Initially, it operates exactly like *BLC* to find $\vec{\rho^*}$ optimal w.r.t. the binary cost J_b^π. Then it finds all sources w for which

$$\rho_w^* < \epsilon R/|W|,$$

sets $\rho_w^* = \epsilon R/|W|$ for each of them, and re-solves the problem over the remaining sources and bandwidth again using *BLC*. Thus, *BLC*ϵ uniformly distributes a small fraction of the bandwidth to sources that would otherwise get no or little crawl rate allocated to it.

The original, optimal *BLC* can be viewed as *BLC*(0). Any $\epsilon > 0$ results in suboptimality w.r.t. J_b^π while ensuring that $J_h^\pi < \infty$. For the experiments, we did a parameter sweep to determine ϵ that resulted in *BLC*ϵ's best performance w.r.t. LAMBDACRAWL's objective J_h^π. **The best value we found for our dataset, and used in all the experiments, is $\epsilon = 0.4$.**

***UniformCrawl* (*UC*)** [9, 23] is a heuristic that assigns an equal crawl rate to all sources:

$$\rho_w = R/|W|.$$

In spite of its simplicity, in our experiments it outperforms *ChangeRateCrawl*, as predicted by Cho & Garcia-Molina [10] (see the caption of Figure 1).

***ChangeRateCrawl* (*CC*)** is the name we give to another heuristic proposed by Cho & Garcia-Molina [10] that sets

$$\rho_w = \frac{\Delta_w R}{\sum_{w' \in W} \Delta_{w'}},$$

thereby crawling sources at a rate proportional to their change rate. Cho & Garcia-Molina [10] pointed out that *ChangeRateCrawl* can be very suboptimal if the set W includes sources that change frequently. This causes *ChangeRateCrawl* to over-commit crawl bandwidth to sources whose changes are near-impossible to keep up with, at the expense of almost ignoring the rest. Our experimental results agree with this observation — *ChangeRateCrawl* turned out to be the weakest-performing algorithm in our experiments.

LAMBDALEARNANDCRAWL (*LLC*), as in Algorithm 4.

LAMBDALEARNANDCRAWLAPPROX (*LLCA*), as in Algorithm 4 with calls to LAMBDACRAWL-INCOMLOBS in LAMBDACRAWL replaced by setting

$$\rho_w = \frac{\mu_w R}{\sum_{w' \in W^-} \mu_{w'}}.$$

per Proposition 9.

***BinaryLambdaLearnAndCrawl(ϵ)* (*BLLC*ϵ)**, the reinforcement learning version of *BLC*ϵ where *BLC*ϵ replaces LAMBDACRAWL in Algorithm 4. This is also the natural RL adaptation of pure *BLC* [2], which otherwise would need a dedicated exploration parameter to ensure data gathering for sources that would get $\rho_w = 0$ under *BLC*'s (currently) optimal policy. As for *BLC*ϵ, we used $\epsilon = 0.4$.

9.4 Experiment 1 (Figure 1)

The goal of this experiment was to assess the harmonic objective J_h^π that we proposed and the binary objective J_b^π widely studied in previous works in terms of robustness: how well do policies optimal w.r.t. one of them behave w.r.t. the other, and vice versa?

In this experiment, we assumed known change rates. To obtain them, we inferred them with estimators in Equations 10 and 11 from our entire 14-week crawl data for 18.5M URLs, and used the resulting estimates as ground truth. Policies were evaluated by plugging in these change rates and policy parameters into the equations in Propositions 1 and 4 and into Equations 12 and 13.

As Figure 1 shows, the harmonic penalty J_h^π we propose is a more flexible choice of objective than the binary penalty J_b^π. LAMBDACRAWL, optimal w.r.t. it, significantly outperforms *BLC*ϵ and *BLC* w.r.t. it, and even the approximate LAMBDACRAWLAPPROX performs at par with *BLC*ϵ according to this objective. What is even more surprising, LAMBDACRAWL also manages to outperform *BLC* on the binary objective J_b^π, for which *BLC* is optimal *if all URLs have only incomplete change observations*. That is, no matter which objective we trust, optimizing for J_h^π yields excellent results.

As a side note, the *UniformCrawl* and *ChangeRateCrawl* heuristics were outperformed by a large margin by the above methods w.r.t. both objectives.

9.5 Experiment 2 (Figure 2)

One of our contributions is a mechanism for taking advantage of complete remote change observations (Algorithm 2). *BLC*ϵ, *BLC*, *UniformCrawl*, and *ChangeRateCrawl* don't have it, and treat all pages as if the only change observations for them came from crawling. While only 4% of web pages in our dataset have an (approximately) complete observation history, can LAMBDACRAWL's and LAMBDACRAWLAPPROX's advantage on them explain the performance gap in experiment 1?

Figure 2 indicates that these URLs are indeed responsible for a significant fraction of LAMBDACRAWL's advantage. In this experiment, we focused only on URLs with complete change observations. Like in the previous experiment, we assumed perfect model knowledge using the previously obtained change rate estimates and estimated policy performance using Propositions 1, 4 and Equations 12 and 13. Treating these URLs as complete-observation URLs, as LAMBDACRAWLCOMPLOBS does, resulted in nearly 5-fold reduction in policy cost, compared to treating these URLs under the conventional change observation model.

Although this gives LAMBDACRAWL an edge over previously proposed techniques, note that even when LAMBDACRAWL treats these URLs conventionally (denoted by the LAMBDACRAWLINCOMLOBS (*LC-IO*) plot in Figure 2), it still noticeably outperforms *BLC* and *BLC*ϵ w.r.t. J_h^π while holding its own against them w.r.t. J_b^π.

9.6 Experiment 3 (Figure 3)

Last but not least, we analyze the reinforcement learning variants of LAMBDACRAWL, LAMBDACRAWLAPPROX, and *BLLC*ϵ. Our motivation for the approximation in Proposition 9 was reducing the number of parameters LAMBDALEARNANDCRAWL has to learn in order to speed up convergence. In this experiment, we explore the tradeoff between the resulting gain in learning speed and the concomitant loss in solution quality.

The evaluation was done in a series of 20 simulated episodes for each algorithm, each episode being executed on a randomly chosen 100,000-URL subsample of the 18.5M URLs (each subsample was used once be each of the three algorithms.) In each episode, we simulated a 21-day run of *LLC*, *LLCA*, and *BLLC*ϵ starting with change rate estimates of 1 change per day for each URL. One epoch (see Algorithm 4) corresponded to 1 day. That is, every (simulated) day each of these algorithms re-estimated the change parameters using the simulated observation data (the simulated data wasn't shared among the algorithms). The simulated data was generated by sampling page changes using the ground truth change rates obtained in previous experiments and sampling page crawls from each algorithm's current policy. At the end of each day, each algorithm reoptimized its policy for the new estimates, and this policy was evaluated using the aforementioned equations *on the ground truth change rates*. For each algorithm, averaged the policy costs for each day across all 20 episodes.

Figure 3 demonstrates that LAMBDALEARNANDCRAWLAPPROX indeed converges quicker than the other algorithms, with its asymptotic performance being better than *BLLC*ϵ's but falling short of LAMBDALEARNANDCRAWL's. *BLLC*ϵ's convergence was the slowest. It could potentially be improved by choosing a larger ϵ at the beginning and gradually "cooling" it. LAMBDALEARNANDCRAWL's advantage, besides convergence speed and asymptotic performance, is that it converges quickly without such parameter tuning. LAMBDALEARNANDCRAWLAPPROX, however, can also be a viable alternative to it, given its simplicity, fast convergence, and good (though suboptimal) policy quality.

10 Proofs

PROOF OF PROPOSITION 1.

First we rearrange Equation 1 as follows, dropping the distributions under the expectation and using W instead of W^- throughout the proof to make the notation less cumbersome:

$$\begin{aligned} J^\pi &= \lim_{T\to\infty} \underset{\substack{CrSeq\sim\pi,\\ ChSeq\sim P(\vec{\Delta})}}{\mathbb{E}} \left[\frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w C(N_w(t))\right) dt\right] \\ &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \mathbb{E}\left[C(N_w(t))\right]\right) dt \end{aligned}$$

Then we use the definition of expectation, chain rule of probabilities, and variable t_{prev} to denote the time when source w was last crawled before time t (although t_{prev} is specific to each source w, for clarity of notation we make this implicit):

$$\begin{aligned} J^\pi &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \mathbb{E}[C(N_w(t))]\right) dt \\ &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \sum_{m=0}^{\infty}\left(C(m)\cdot\mathbb{P}[N_w(t)=m]\right)\right) dt \\ &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \sum_{m=0}^{\infty}\left(C(m)\int_0^t \mathbb{P}[N_w(t)=m \mid t-t_{prev}=T']\,\mathbb{P}[t-t_{prev}=T']dT'\right)\right) dt \end{aligned}$$

Now using the fact that our policy is a set of Poisson processes with parameters ρ_w and page changes are governed by another set of Poisson processes with parameters Δ_w, we can plug in appropriate expressions for the probabilities:

$$\begin{aligned} J^\pi &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \sum_{m=0}^{\infty}\left(C(m)\int_0^t \left(\frac{(T'\Delta_w)^m e^{-T'\Delta_w}}{m!}\right)\left(\rho e^{-\rho_w T'}\right)dT'\right)\right) dt \\ &= \lim_{T\to\infty} \frac{1}{T}\int_0^T \left(\sum_{w\in W}\mu_w \sum_{m=0}^{\infty}\left(\frac{C(m)\rho_w\Delta_w^m}{m!}\int_0^t T'^m e^{-(\rho_w+\Delta_w)T'}dT'\right)\right) dt \end{aligned}$$

We do a variable substitution $u=(\rho_w+\Delta_w)T'$, so $T'=\frac{u}{\Delta_w+\rho_w}$ and $dT'=\frac{du}{\Delta_w+\rho_w}$:

$$J^\pi = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\sum_{w\in W} \mu_w \sum_{m=0}^{\infty} \left(\frac{C(m)\rho_w \Delta_w^m}{m!} \int_0^t \left(\frac{u}{\Delta_w + \rho_w} \right)^m e^{-u} \frac{du}{\Delta_w + \rho_w} \right) \right) dt$$
$$= \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\sum_{w\in W} \mu_w \sum_{m=0}^{\infty} \left(\frac{C(m)}{m!} \left(\frac{\rho_w}{\Delta_w + \rho_w} \right) \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \int_0^t u^m e^{-u} du \right) \right) dt$$

Consider $F(m,t) = \int_0^t u^m e^{-u} du$. By definition of gamma functions, $F(m,t) = \Gamma(m+1) - \Gamma(m+1,t) = m! - \Gamma(m+1,t)$. Recalling that $C(m) = H(m)$ for $m > 0$ and $C(0) = 0$ (Equation 2), we get

$$J^\pi = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\sum_{w\in W} \mu_w \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\rho_w}{\Delta_w + \rho_w} \right) \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m (m! - \Gamma(m+1,t)) \right) \right) dt$$
$$= \lim_{T\to\infty} \frac{1}{T} \Bigg(\sum_{w\in W} \left(\frac{\mu_w \rho_w}{\Delta_w + \rho_w} \right) \Bigg(\int_0^T \sum_{m=1}^{\infty} \left(H(m) \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \right) dt -$$
$$- \int_0^T \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \Gamma(m+1,t) \right) dt \Bigg) \Bigg) \qquad (14)$$

Now consider for each $w \in W$ functions $G(T) = \int_0^T \sum_{m=1}^{\infty} \left(H(m) \left(\frac{\Delta_w}{\Delta_w+\rho_w} \right)^m \right) dt$, $R(t) = \int_0^T \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w+\rho_w} \right)^m \Gamma(m+1,t) \right) dt$. Consider $\lim_{T\to\infty} \frac{1}{T}(G(T) - R(T))$ so that

$$J^\pi = \sum_{w\in W} \left(\mu_w \left(\frac{\rho_w}{\Delta_w + \rho_w} \right) \left(\lim_{T\to\infty} \frac{1}{T} (G(T) - R(T)) \right) \right) \qquad (15)$$

Thus, if $\lim_{T\to\infty} \frac{1}{T}(G(T) - R(T))$ exists, is finite, and we can compute it, we can compute J^π as well. Focusing on $G(T)$ and recalling that $\sum_{m=1}^{\infty} H(m)x^m = -\frac{\ln(1-x)}{1-x}$ for $|x| < 1$, we see that $G(T) = -\frac{\ln(\frac{\rho_w}{\Delta_w+\rho_w})}{\frac{\rho_w}{\Delta_w+\rho_w}} T$, so $\lim_{T\to\infty} \frac{G(T)}{T} = -\frac{\ln(\frac{\rho_w}{\Delta_w+\rho_w})}{\frac{\rho_w}{\Delta_w+\rho_w}}$ exists. Focusing on $R(T)$, we see that since $m! < \Gamma(m+1,t)$ for any $t > 0$ and since $R(T)$ is an integral of a non-negative function, we have $0 \le R(T) \le G(T) < \infty$, so $\lim_{T\to\infty} \frac{R(T)}{T}$ exists as well. Therefore, we can write $\lim_{T\to\infty} \frac{1}{T}(G(T) - R(T)) = \lim_{T\to\infty} \frac{G(T)}{T} - \lim_{T\to\infty} \frac{R(T)}{T}$.

We already know $\lim_{T\to\infty} \frac{G(T)}{T}$. To evaluate $\lim_{T\to\infty} \frac{R(T)}{T}$, we upper-bound $R(T)$. Again using the fact that it is an integral of a non-negative function, we have

$$R(T) = \int_0^T \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \Gamma(m+1,t) \right) dt$$
$$< \int_0^\infty \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \Gamma(m+1,t) \right) dt$$
$$= \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \int_0^\infty \Gamma(m+1,t) dt \right)$$
$$= \sum_{m=1}^{\infty} \left(\frac{H(m)}{m!} \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \Gamma(m+2) \right)$$
$$= \sum_{m=1}^{\infty} \left((m+1)H(m) \left(\frac{\Delta_w}{\Delta_w + \rho_w} \right)^m \right)$$

We have used the fact that $\int_0^\infty \Gamma(m+1,t)dt = \Gamma(m+2) = (m+1)!$. The series $\sum_{m=1}^\infty (m+1)H(m)x^m$ converges for $|x|<1$ to some limit $L>0$, so we have $\lim_{T\to\infty}\frac{R(T)}{T} \leq \lim_{T\to\infty}\frac{L}{T} = 0$ and $\lim_{T\to\infty}\frac{1}{T}(G(T)-R(T)) = -\frac{\ln(\frac{\rho_w}{\Delta_w+\rho_w})}{\frac{\rho_w}{\Delta_w+\rho_w}}$. Plugging this back into Equation 15, we get

$$J^\pi = -\sum_{w\in W}\mu_w \ln\left(\frac{\rho_w}{\Delta_w+\rho_w}\right)$$

■

Corollary 1. *Under the harmonic penalty $C(n)$ (Equation 2), any policy that crawls each source at a fixed Poisson rate ρ_w and assigns $\rho_w > 0$ to each w with $\mu_w, \Delta_w > 0$ is strictly preferable to any such policy that assigns $\rho_w = 0$ to any such source.*

Proof. Proposition 1 implies that any policy π with $\rho_w = 0$ for any source w with $\mu_w, \Delta_w > 0$ has $J^\pi = \infty$, whereas any π with $\rho_w > 0$ for every source w with $\mu_w, \Delta_w > 0$ has $J^\pi < \infty$ ■

PROOF OF PROPOSITION 2. The high-level idea is to apply the method of Lagrange multipliers to Problem 1's relaxation *without* inequality constraints $\vec{\rho} \geq 0$ and show that (a) only one local maximum of this relaxation is within the region given by $\vec{\rho} \geq 0$ – the one satisfying Equation system 5 – and (b) solutions that touch the boundary of this region, i.e., have $\rho_w = 0$ for any $w \in W^-$, are suboptimal. In fact, part (b) follows immediately from Corollary 1, so we only need to solve the relaxation and show part (a).

To apply the method of Lagrange multipliers to the relaxation, we set $f(\vec{\rho}) = \overline{J}^\pi = \sum_{w\in W^-}\mu_w \ln\left(\frac{\rho_w}{\Delta_w+\rho_w}\right)$ and $g(\vec{\rho}) = \sum_{w\in W^-}\rho_w - R$. We need to solve

$$\begin{cases}\nabla f(\vec{\rho}) = \lambda \nabla g(\vec{\rho})\\ g(\vec{\rho}) = 0.\end{cases}$$

For any $w \in W^-$, we have $\frac{\partial g}{\partial \rho_w} = 1$ and $\frac{\partial f}{\partial \rho_w} = \mu_w \frac{\Delta_w+\rho_w}{\rho_w}\frac{\Delta_w}{(\Delta_w+\rho_w)^2} = \frac{\mu_w\Delta_w}{\Delta_w\rho_w+\rho_w^2}$, so the above system of equations turns into

$$\begin{cases}\frac{\mu_w\Delta_w}{\Delta_w\rho_w+\rho_w^2} = \lambda, & \text{for all } w\in W^-\\ \sum_{w\in W^-}\rho_w = R\end{cases}$$

and therefore

$$\begin{cases}\lambda\rho_w^2 + \lambda\Delta_w\rho_w - \mu_w\Delta_w = 0, & \text{for all } w\in W^-\\ \sum_{w\in W^-}\rho_w = R\end{cases}$$

Solving each quadratic equation separately, we get

$$\begin{cases}\rho_w = \frac{-\Delta_w \pm\sqrt{\Delta_w^2 + \frac{4\mu_w\Delta_w}{\lambda}}}{2}, & \text{for all } w\in W^-\\ \sum_{w\in W^-}\rho_w = R\end{cases}$$

This gives all potential solutions to the relaxation of Problem 1. Now consider the inequality constraints $\vec{\rho} \geq 0$ omitted so far. Observe that any real solution to the above system that has $\rho_w = \frac{-\Delta_w - \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2}$ for any $w \in W^-$ implies $\rho_w < 0$ for $\mu_w, \Delta_w > 0$, which violates these constraints. Therefore, any solution to Problem 1 itself must satisfy

$$\begin{cases} \rho_w = \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2}, & \text{for all } w \in W^- \\ \sum_{w \in W^-} \rho_w = R \end{cases}$$

and have $\lambda \geq 0$ (otherwise $\vec{\rho} < 0$, again violating the inequality constraints). Although the first group of equations are non-linear, note that each $\rho_w(\lambda)$ is strictly monotone decreasing in λ for $\lambda \geq 0$, so $\sum_{w \in W^-} \rho_w$ is strictly monotone decreasing too implying that $\sum_{w \in W^-} \rho_w(\lambda) = R$ has a unique solution in λ, and therefore there is a unique $\vec{\rho}$ satisfying the above system of equations. Thus, this $\vec{\rho}$ is the solution to Problem 1 and therefore corresponds to $\pi^* \in \Pi^-$. ■

PROOF OF PROPOSITION 3

We start by combining Equation system 5,

$$\begin{cases} \rho_w = \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2}, & \text{for all } w \in W^- \\ \sum_{w \in W^-} \rho_w = R, \end{cases} \tag{16}$$

into one equation:

$$\sum_{w \in W^-} \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2} = R$$

Bisection search is guaranteed to converge to *some* solution λ of this equation as long as we initialize the search with $\lambda_{lower}, \lambda_{upper}$ s.t. $\lambda \in [\lambda_{lower}, \lambda_{upper}]$. However, all $\lambda_- \leq 0$ that solve Equation system 5 correspond to solutions that have $\vec{\rho} < 0$. At the same time, from the proof of Proposition 2 we know that there is exactly one $\lambda_+ > 0$ that solves Equation system 5, and it corresponds to the (unique) the optimal solution $\vec{\rho}^*$ of Problem 1. We want bisection search to find only this $\lambda_+ > 0$, so we want $\lambda_{lower}, \lambda_{upper}$ s.t. $\lambda_+ \in [\lambda_{lower}, \lambda_{upper}]$ *and* $\lambda_{lower}, \lambda_{upper} > 0$.

To find these bounds, we observe that the l.h.s. of the above equation is monotonically decreasing in λ for $\lambda > 0$, so if we find any $\lambda_l > 0$ that guarantees

$$\sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda_l}} \geq \Delta_w + \frac{2R}{|W|} \text{ for all } w \in W^- ,$$

then we have

$$\sum_{w \in W^-} \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda_l}}}{2} \geq R$$

and hence $\lambda_l \leq \lambda_+$. To find such λ_l, we perform a series of algebraic manipulations:

$$
\begin{aligned}
&\sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda_l}} \geq \Delta_w + \frac{2R}{|W|} \text{ for all } w \in W^-, \lambda_l > 0 \\
\Longleftrightarrow & \Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda_l} \geq \left(\Delta_w + \frac{2R}{|W^-|}\right)^2 \text{ for all } w \in W^-, \lambda_l > 0 \\
\Longleftrightarrow & \frac{4\mu_w \Delta_w}{\lambda_l} \geq \frac{4\Delta_w R}{|W^-|} + 4\left(\frac{R}{|W^-|}\right)^2 \text{ for all } w \in W^-, \lambda_l > 0 \\
\Longleftrightarrow & \lambda_l \leq \frac{|W^-|^2 \mu_w \Delta_w}{|W^-|\Delta_w R + R^2} \text{ for all } w \in W^-, \lambda_l > 0 \\
\Longleftarrow & \lambda_l \leq \frac{|W^-|^2 \min_{w \in W^-}\{\mu_w\} \min_{w \in W^-}\{\Delta_w\}}{|W^-| \max_{w \in W^-}\{\Delta_w\} R + R^2}, \lambda_l > 0
\end{aligned}
$$

Since the r.h.s. of this inequality is always positive, we set $\lambda_{lower} = \frac{|W^-|^2 \min_{w \in W^-}\{\mu_w\} \min_{w \in W^-}\{\Delta_w\}}{|W^-| \max_{w \in W^-}\{\Delta_w\} R + R^2}$. An analogous chain of reasoning shows that we can choose $\lambda_{upper} = \frac{|W^-|^2 \max_{w \in W^-}\{\Delta_w\} \max_{w \in W^-}\{\mu_w\}}{|W^-| \min_{w \in W^-}\{\Delta_w\} R + R^2}$.

To establish a bound on the running time, we observe that each iteration of LAMBDACRAWL-INCOMLOBS involves evaluating $\sum_{w \in W^-} \rho_w$, which takes $O(|W^-|)$ time, so by the properties of bisection search the total running time is $O(\log_2(\frac{\lambda_{upper} - \lambda_{lower}}{\epsilon})|W^-|)$. ■

PROOF OF PROPOSITION 4 Let $Ch_w(t)$ denote the total number of changes that have happened at source w in time interval $[0, t]$. As in the proof of Proposition 1, we start with rearranging the cost function in Equation 1, dropping the distributions under the expectation and using W instead of W^o throughout the proof to make the notation less cumbersome. Using the definition of expectation and chain rule of probabilities to get:

$$
\begin{aligned}
J^\pi &= \lim_{T \to \infty} \mathop{\mathbb{E}}_{\substack{CrSeq \sim \pi, \\ ChSeq \sim P(\vec{\Delta})}} \left[\frac{1}{T} \int_0^T \left(\sum_{w \in W} \mu_w C(N_w(t)) \right) dt \right] \\
&= \lim_{T \to \infty} \frac{1}{T} \int_0^T \left(\sum_{w \in W} \mu_w \mathbb{E}\left[C(N_w(t))\right] \right) dt \\
&= \lim_{T \to \infty} \frac{1}{T} \int_0^T \left(\sum_{w \in W} \mu_w \sum_{m=0}^{\infty} \left(C(m) \cdot \mathbb{P}[N_w(t) = m] \right) \right) dt \\
&= \lim_{T \to \infty} \frac{1}{T} \int_0^T \left(\sum_{w \in W} \mu_w \sum_{c=0}^{\infty} \sum_{m=0}^{\infty} \left(C(m) \cdot \mathbb{P}[N_w(t) = m | Ch_w(t) = c] \mathbb{P}[Ch_w(t) = c] \right) \right) dt
\end{aligned}
$$

Since changes at every source w are governed by a Poisson process with rate Δ_w, $\mathbb{P}[Ch_w(t) = c] = \frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!}$. Now, consider $\mathbb{P}[N_w(t) = m | Ch_w(t) = c]$. Recall that whenever source $w \in W^o$ changes, we find out about the change immediately; with probability p_w the policy $\pi \in \Pi^o$ then crawls source w straight away, and with probability $(1 - p_w)$ it waits to make this decision until we find out about w's next change. Therefore, the only way we can have $N_w(t) = m$ is if our policy crawled source w $m + 1$ changes ago *and* has *not* chosen to crawl w after any of the m changes that happened since, or it hasn't chosen to crawl w since "the beginning of time". Thus, $N_w(t)$ is geometrically distributed, assuming that at least $m + 1$ changes actually happened at source w in the time interval $[0, 1]$. Thus, we have

$$\mathbb{P}[N_w(t) = m | Ch_w(t) = c] = \begin{cases} p_w(1-p_w)^m, & \text{if } c \geq m+1 \\ (1-p_w)^m, & \text{if } c = m \\ 0 & \text{otherwise} \end{cases}$$

Recalling that $C(m) = H(m)$ for $m > 0$ and $C(0) = 0$ (Equation 2) and putting everything together, we have

$$\begin{aligned} J^\pi &= \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \sum_{m=0}^{\infty} \Big(C(m) \cdot \mathbb{P}[N_w(t) = m | Ch_w(t) = c] \mathbb{P}[Ch_w(t) = c] \Big) \right) dt \\ &= \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=1}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=1}^{c-1} H(m)(1-p_w)^m + H(c)(1-p_w)^c \right) \right) dt \\ &= \lim_{T\to\infty} \frac{1}{T} \Bigg(\int_0^T \sum_{w\in W} \mu_w \sum_{c=1}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=1}^{\infty} H(m)(1-p_w)^m \right) dt \\ &\qquad - \int_0^T \sum_{w\in W} \mu_w \sum_{c=1}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} H(m)(1-p_w)^m \right) dt \\ &\qquad + \int_0^T \sum_{w\in W} \mu_w \sum_{c=1}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \Big(H(c)(1-p_w)^c \Big) dt \Bigg) \end{aligned}$$

Now consider

$$\begin{aligned} G(T) &= \int_0^T \sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=0}^{\infty} C(m)(1-p_w)^m \right) dt \\ R(T) &= \int_0^T \sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) dt \\ F(T) &= \int_0^T \sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \Big(C(c)(1-p_w)^c \Big) dt \end{aligned}$$

Since

$$J^\pi = \lim_{T\to\infty} \frac{1}{T} (G(T) - R(T) + F(T)),$$

if we show that each of $\lim_{T\to\infty} \frac{G(T)}{T}$, $\lim_{T\to\infty} \frac{R(T)}{T}$, and $\lim_{T\to\infty} \frac{F(T)}{T}$ exists and manage to compute them, then we will know J^π. The rest of the proof focuses on computing these limits.

Consider $G(T)$. Note that $p_w \sum_{m=0}^{\infty} C(m)(1-p_w)^m = p_w \sum_{m=1}^{\infty} H(m)(1-p_w)^m$ doesn't depend on c. We can use the identity $\sum_{m=1}^{\infty} H(m)x^m = -\frac{\ln(1-x)}{1-x}$ for $|x| < 1$ to get $p_w \sum_{m=0}^{\infty} C(m)(1-p_w)^m = -\ln(p_w)$, so $G(T) = -\int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \ln(p_w) \right) dt$. To simplify $G(T)$ further, note that $\sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right)$ is just the probability of *any* number of changes occurring in time interval $[0, t]$ under a Poisson process, and therefore equals 1. Thus, $G(T) = -\int_0^T \left(\sum_{w\in W} \mu_w \ln(p_w) \right) dt = -T \sum_{w\in W} \mu_w \ln(p_w)$, so

$$\lim_{T\to\infty} \frac{G(T)}{T} = -\sum_{w\in W} \mu_w \ln(p_w).$$

Consider $R(T)$. Observe that $C(m) = H(m) < m$ for $m > 1$, so $p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m < p_w \sum_{m=c}^{\infty} m(1-p_w)^m = \frac{(1-p_w)^c(cp_w - p_w + 1)}{p_w}$. Because $(1-p_w)^c$ decreases in c much faster than $\frac{1}{c(cp_w - p_w + 1)}$ for any fixed $0 < p_w \leq 1$, for any $w \in W$ there is a c^*_w s.t. $p_w \sum_{m=c}^{\infty} C(m)(1-$

$p_w)^m < \frac{(1-p_w)^c(cp_w - p_w + 1)}{p_w} < \frac{1}{c}$ for any $c > c_w^*$. Let $c^* = \max_{w\in W} c_w^*$. We can then upper-bound $R(T)$ as follows:

$$R(T) = \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) \right) dt = R_1(T) + R_2(T),$$

where

$$R_1(T) = \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{c^*-1} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) \right) dt$$

$$R_2(T) = \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=c^*}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) \right) dt$$

Considering $R_1(T)$, recalling that $C(m) = H(m)$ for $m > 0$, we have $p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \leq p_w \sum_{m=1}^{\infty} C(m)(1-p_w)^m = -\ln(p_w)$, as shown previously. Also, for any c we have $\int_0^T \frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} dt = \frac{\Gamma(c+1)-\Gamma(c+1,\Delta_w T)}{\Delta_w c!}$. Therefore,

$$\begin{aligned} R_1(T) &= \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{c^*-1} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) \right) dt \\ &< -\sum_{w\in W} \mu_w \ln(p_w) \sum_{c=0}^{c^*-1} \frac{\Gamma(c+1) - \Gamma(c+1, \Delta_w T)}{\Delta_w c!} \end{aligned}$$

Since $\lim_{T\to\infty} \frac{\Gamma(c+1,\Delta_w T)}{T} = 0$, $R_1(T)$ is therefore upper-bounded by a finite sum of terms that all go to 0 when divided by T as $T \to \infty$. Since $R_1(T) \geq 0$ also holds, we have $\lim_{T\to\infty} \frac{R_1(T)}{T} = 0$.

Considering $R_2(T)$, we use our definition of c^* to write

$$\begin{aligned} R_2(T) &= \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=c^*}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(p_w \sum_{m=c}^{\infty} C(m)(1-p_w)^m \right) \right) dt \\ &< \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=c^*}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{c!} \right) \left(\frac{1}{c} \right) \right) dt = \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=c^*}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{(c+1)!} \right) \right) dt \\ &\leq \int_0^T \left(\sum_{w\in W} \mu_w \sum_{c=0}^{\infty} \left(\frac{e^{-\Delta_w t}(\Delta_w t)^c}{(c+1)!} \right) \right) dt \\ &= \int_0^T \left(\sum_{w\in W} \mu_w \left(\frac{1 - e^{-\Delta_w t}}{\Delta_w t} \right) \right) dt \\ &= \sum_{w\in W} \mu_w \frac{\ln(\Delta_w T) + \Gamma(0, \Delta_w T) + \gamma}{\Delta_w}, \end{aligned}$$

where γ is the Euler-Mascheroni constant. Since $\lim_{T\to\infty} \frac{\Gamma(0,\Delta_w T)}{T} = 0$, $R_2(T)$ is therefore upper-bounded by a finite sum of terms that all go to 0 when divided by T as $T \to \infty$. Since $R_2(T) \geq 0$ also holds, we have $\lim_{T\to\infty} \frac{R_2(T)}{T} = 0$. Thus, we have

$$\lim_{T\to\infty} \frac{R(T)}{T} = \lim_{T\to\infty} \frac{R_1(T) + R_2(T)}{T} = \lim_{T\to\infty} \frac{R_1(T)}{T} + \lim_{T\to\infty} \frac{R_2(T)}{T} = 0$$

Consider $F(T)$. Observe that for a suitably chosen constant $s > 0$, $sR(T) > F(T)$. Therefore, since $\lim_{T\to\infty} \frac{R(T)}{T} = 0$, $\lim_{T\to\infty} \frac{F(T)}{T} = 0$ too.

We have thus shown that

$$J^{\pi} = \lim_{T\to\infty} \frac{G(T) - R(t) + F(T)}{T} = \lim_{T\to\infty} \frac{G(T)}{T} - \lim_{T\to\infty} \frac{R(T)}{T} + \lim_{T\to\infty} \frac{F(T)}{T} = -\sum_{w\in W} \mu_w \ln(p_w)$$

■

PROOF OF PROPOSITION 5. Since under any $\vec{p} \geq 0$ crawl rates $\vec{\rho}$ are related to crawl probabilities via $\rho_w = p_w \Delta_w$, to apply the method of Lagrange multipliers to the relaxation of Problem 2 that takes into account only the bandwidth constraint we set $f(\vec{p}) = \overline{J}^{\pi} = \sum_{w\in W^o} \mu_w \ln(p_w)$ and $g(\vec{\rho}) = \sum_{w\in W^o} p_w \Delta_w - R$. We need to solve

$$\begin{cases} \nabla f(\vec{p}) = \lambda \nabla g(\vec{p}) \\ g(\vec{p}) = 0. \end{cases}$$

For any $w \in W^o$, we have $\frac{\partial g}{\partial p_w} = \Delta_w$ and $\frac{\partial f}{\partial p_w} = \frac{\mu_w}{p_w}$, so the above system of equations turns into

$$\begin{cases} \frac{\mu_w}{p_w} = \lambda \Delta_w, & \text{for all } w \in W^o \\ \sum_{w\in W^o} p_w \Delta_w = R \end{cases}$$

and therefore

$$\begin{cases} p_w = \frac{R\mu_w}{\Delta_w \sum_{w\in W^o} \mu_w} \text{for all } w \in W^o \\ \lambda = \frac{\sum_{w\in W^o} \mu_w}{R} \end{cases}$$

This is the only solution yielded by the method of Lagrange multipliers, so it is the unique maximizer of the relaxation. ■

PROOF OF PROPOSITION 6.

To establish LAMBDACRAWL-COMPLOBS's correctness, we first prove the following lemma, which establishes that any source w that violates its $p_w \leq 1$ constraint in any iteration of LAMBDACRAWL-COMPLOBS must have $p^*_w = 1$:

Lemma 1. *Let $\vec{p}^*$ be the maximizer of Problem 2, and let $\vec{p}'^*$ be the maximizer of the relaxation Problem 2 with the same inputs but with inequality constraints ignored. Then any source w that has $\vec{p}'^*$ has $p'^*_w > 1$, violating its inequality constraint, necessarily has $p^*_w = 1$.*

Proof. For convenience, we rewrite Problem 2 as an equivalent problem of maximizing $\overline{J}_{mod} = \sum_{w\in W^o} \mu_w \ln(\rho_w)$ under the constraints $\sum_{w\in W^o} \rho_w = R$ and $\rho_w \leq \Delta_w$ for every $w \in W^o$. Consider its relaxation with $\rho_w \leq \Delta_w$ for every $w \in W^o$ ignored. By the equivalent of Proposition 5 for this reformulation, $\vec{\rho}'^*$ is unique and must have $p'^*_w = \rho'^*_w/\Delta_w$ for each source w. The Lagrangian of $\mathcal{L}(\vec{\rho}, \lambda_0) := \overline{J}_{mod}(\vec{\rho}) - \lambda_0(\sum_{w\in W^o} \rho_w - R)$ must have $\nabla\mathcal{L}(\vec{\rho}'^*, \lambda'^*_0) = 0$ for the optimal solution $(\vec{\rho}'^*, \lambda'^*_0)$ (which, again, encodes the optimal p'^*_w for the relaxation of the original formulation of Problem 2 via $p'^*_w = \rho'^*_w/\Delta_w$). From this we see that $\frac{\partial}{\partial\rho_u}\overline{J}_{mod}\restriction_{\rho'^*_u} = \lambda'^*_0 = \frac{\partial}{\partial\rho_w}\overline{J}_{mod}\restriction_{\rho'^*_w}$, for any sources $u, w \in W^o$. Since $\frac{\partial}{\partial\rho_w}\overline{J}_{mod} = \frac{\mu_w}{\rho_w}$, this implies

$$\frac{\mu_u}{\rho'^*_u} = \frac{\mu_w}{\rho'^*_w} \quad \text{for all } u, w \in W^o\,. \tag{17}$$

Consider the slack-variable formulation of Problem 2 with slack variables $\{q_w\}_{w \in W^o}$. Inequality constraints in this formulation turn into $\rho_w = \Delta_w - q_w$ for $q_w \geq 0$. In this formulation, we now have

$$\mathcal{L}(\vec{\rho}, \lambda_0, \lambda_1, \ldots, \lambda_{|W^o|}) := \mathcal{L}(\vec{\rho}, \lambda_0) - \sum_{w \in W^o} \lambda_w(\rho_w - \Delta_w) .$$

where, by the Karush-Kuhn-Tucker conditions, $\lambda_w \geq 0$ for all w. By complementary slackness, $\lambda_w^* = 0$ for every $w \in W^o$ such that $q_w > 0$, i.e., for every w that does *not* activate its inequality constraint, under the optimal solution $(\vec{\rho}^*, \lambda_0^*, \lambda_w^*, \ldots, \lambda_{|W^o|}^*)$. This implies that $\frac{\partial}{\partial \rho_u} \overline{J}_{mod} \restriction_{\rho_u^*} -\lambda_u^* = \lambda_0^* = \frac{\partial}{\partial \rho_w} \overline{J}_{mod} \restriction_{\rho_w^*} -\lambda_w^*$, for any sources $u, w \in W^o$, i.e.,

$$\frac{\mu_u}{\rho_u'^*} - \lambda_u^* = \lambda_0^* = \frac{\mu_w}{\rho_w'^*} - \lambda_w^* \quad \text{for all } u, w \in W^o . \tag{18}$$

Now, suppose for contradiction that there is a source $u \in W^o$ that has $p_u'^* > 1$ but $p_u^* < 1$, implying that $\rho_u'^* > \Delta_u$ but $\rho_u^* < \Delta_u$. This, in turn, implies that (a) $\rho_u'^* > \rho_u^*$ and (b) $\lambda_u^* = 0$, since u doesn't activate its inequality constraint under $\vec{\rho}^*$ (and hence under $\vec{p}^*$). Then, since $\sum_{v \in W^o} \rho_v^* = \sum_{v \in W^o} \rho_v'^* = R$, there must also exist some other source $w \neq u$ such that $\rho_w'^* < \rho_w^*$. For this source, $\lambda_w^* \geq 0$, so $\lambda_w^* \geq \lambda_u^*$.

Recall that $\overline{J}_{mod}$ is strictly concave, and its partial derivatives $\frac{\mu_v}{\rho_v}$ are monotone decreasing in every non-negative ρ_v. Together with $\rho_u'^* > \rho_u^*$, $\rho_w'^* < \rho_w^*$, $\lambda_w^* \geq \lambda_u^*$, and Equation 18, this implies

$$\frac{\mu_u}{\rho_u'^*} < \frac{\mu_u}{\rho_u^*} \leq \frac{\mu_w}{\rho_w^*} < \frac{\mu_w}{\rho_w'^*}$$

But this contradicts Equation 17, completing the proof of the lemma. ■

The optimality of LAMBDACRAWL-COMPLOBS now follows by induction. Its every iteration except the last one identifies at least one constraint that is active under $\vec{p}^*$, by the above lemma, and thereby assigns an optimal p_w^* to some sources, leaving optimal crawl probabilities for others to be found in subsequent iterations. The solution for the sources remaining in the final iteration, which does not violate any inequality constraints, is optimal by Proposition 5. Therefore, LAMBDACRAWL-COMPLOBS arrives at the optimal solution, and that solution is unique because Problem 2's maximization objective is concave as a sum of concave functions, and the optimization region is convex.

We note that the proof so far is similar to the proof of Lemma 3.3 from [21] for a different concave function F under constraints of the form $x_1 + \ldots + x_k \leq c_k$, where $x_1, \ldots, x_k$ is a subset of F's variables and c_k is a constraint.

Since in each iteration LAMBDACRAWL-COMPLOBS removes at least one source from further consideration, it makes at most $|W^o|$ iterations. In each iteration it applies Proposition 5, which takes $O(|W^o|)$ time, yielding the overall time complexity of $O(|W^o|^2)$. ■

PROOF OF PROPOSITION 7. See the paper. ■

PROOF OF PROPOSITION 8. LAMBDALEARNANDCRAWL starts with strictly positive finite estimates $\hat{\vec{\Delta}}_0$ of change rates. Since LAMBDACRAWL, which LAMBDALEARNANDCRAWL uses for determining crawl rates for the next epoch, is optimal to any desired precision (Proposition 7), it follows from Corollary 1 that it returns positive $\vec{\rho}_1^*, \vec{p}_1^* > 0$, and $0 < \vec{\mu}, R < \infty$ guarantees that these crawl rates are also finite. In subsequent iterations, $\hat{\vec{\Delta}}_n$ are estimated using Equations 10 and 11 with smoothing terms (lines 11 and 13 of Algorithm 4), and the smoothing terms ensure that the change rate estimates are finite and bounded away from 0: $0 < \delta_{min} \leq \hat{\vec{\Delta}}_n < \infty$, where δ_{min} is implied by the aforementioned estimators and specific smoothing term values. This, along with finite positive $\vec{\mu}$

and R, ensures that $0 < \vec{\hat{\rho}}^*_{n+1}, \vec{\hat{p}}^*_{n+1} < \delta_{max}$. Hence, by induction, no source is ever starved, and no source is crawled infinitely frequently. This ensures, together with consistency of estimators from Equations 10 and 11, that if at least the last iteration N_{epoch} uses the entire observation history, i.e. $S(N_{epoch}) = length(obs\_hist)$, then change rate estimates converge to the true change rates in probability: $\text{plim}_{N_{epochs} \to \infty} \vec{\hat{\Delta}}_{N_{epochs}} = \vec{\Delta}$, as long as $\vec{\Delta}$ doesn't change with time. Optimality of LAMBDACRAWL then implies probabilistic convergence of $(\vec{\hat{\rho}}^*_{N_{epochs}}, \vec{\hat{p}}^*_{N_{epochs}})$ to $(\vec{\hat{\rho}}^*, \vec{\hat{p}}^*)$ as well. ■

PROOF OF PROPOSITION 9. According to Equation system 5, the parameter vector $\vec{\rho}$ of the optimal $\pi^* \in \Pi^-$ that minimizes the expected harmonic penalty in the absence of remote change observations (Problem 1) satisfies

$$\begin{cases} \rho_w = \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2}, & \text{for all } w \in W^- \\ \sum_{w \in W^-} \rho_w = R \end{cases}$$

If $\frac{\mu_w}{\Delta_w} = c$ for all $w \in W^-$, $c > 0$, then we can express $\Delta_w = c' \mu_w$ where $c' = 1/c$ and plug it into the above equations to get

$$\begin{aligned} \rho_w &= \frac{-\Delta_w + \sqrt{\Delta_w^2 + \frac{4\mu_w \Delta_w}{\lambda}}}{2} \\ &= \frac{-c'\mu_w + \sqrt{(c'\mu_w)^2 + \frac{4c'\mu_w^2}{\lambda}}}{2} \\ &= \frac{-c'\mu_w + \mu_w \sqrt{\frac{c'^2\lambda + 4c'}{\lambda}}}{2} \\ &= \mu_w \left(\frac{-c' + \sqrt{\frac{c'^2\lambda + 4c'}{\lambda}}}{2} \right) \quad \text{for all } w \in W^- \end{aligned} \tag{19}$$

Plugging this into the remaining equation from the above system, $\sum_{w \in W^-} \rho_w = R$, we get

$$\begin{aligned} \sum_{w \in W^-} \mu_w \left(\frac{-c' + \sqrt{\frac{c'^2\lambda + 4c'}{\lambda}}}{2} \right) = R \implies \\ \frac{-c' + \sqrt{\frac{c'^2\lambda + 4c'}{\lambda}}}{2} &= \frac{R}{\sum_{w \in W^-} \mu_w} \implies \\ \frac{c'^2\lambda + 4c'}{\lambda} &= \left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 \implies \\ \lambda \left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2\lambda = 4c' \implies \\ \lambda &= \frac{4c'}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2} \end{aligned}$$

Plugging this and $\Delta_w = c'\mu_w$ back into Equations 19, we get for all $w \in W^-$

$$
\begin{aligned}
\rho_w &= \mu_w \left(\frac{-c' + \sqrt{\dfrac{c'^2 \left(\frac{4c'}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2} \right) + 4c'}{\frac{4c'}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2}}}}{2} \right) \\
&= \mu_w \left(\frac{-c' + \sqrt{\dfrac{4c' \left(\left(\frac{c'^2}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2} \right) + 1 \right)}{\frac{4c'}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2}}}}{2} \right) \\
&= \mu_w \left(\frac{-c' + \sqrt{\dfrac{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 \left(\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2 \right)}{\left(\frac{2R}{\sum_{w \in W^-} \mu_w} + c' \right)^2 - c'^2}}}{2} \right) \\
&= \mu_w \left(-c' + \frac{R}{\sum_{w \in W^-} \mu_w} + c' \right) \\
&= \frac{\mu_w R}{\sum_{w \in W^-} \mu_w}
\end{aligned}
$$

■