Optimal Crawling Strategies for Web Search Engines

Jay Sethuraman
IEOR Department, Columbia University

joint work with

Joel L. Wolf, Mark S. Squillante and Philip S. Yu
IBM T. J. Watson Research Center

Leyla Ozsen
Northwestern University

May 8, 2002

Introduction

- Web search engines employ multiple crawlers
  - To maintain local copies of web pages
  - To build data structures such as inverted indexes
- But web pages are updated frequently
  - 23% of web pages change daily
  - 40% of commercial web pages change daily
  - Half-life for a web page is 10 days
- So crawlers must revisit web pages frequently to maintain *freshness*
  - Question: How should one do this optimally?

Two part Scheme

- Component 1: Solve the “Crawling frequency” problem
  - Find optimal number of times to crawl each page
  - Find optimal times to crawl each page
- Component 2: Solve the “Crawler Scheduling” problem
  - Create optimal achievable crawler schedule based on idealized crawler times

Contributions

- Crawling frequency problem
  - More appropriate optimization metric
    - Based on level of *embarrassment*, not just on staleness
  - Unified framework (Stationary Marked Point Processes) to handle wide variety of web page update distribution types
    - Poisson, Pareto, Weibull
    - Quasi-deterministic
  - State-of-art algorithms for finding optimal number of crawls
    - General and practical: Real-life constraints
    - Extraordinary computationally efficient: Problems are *huge*
  - Algorithms for finding ideal crawl times

Contributions

- Scheduling problem
  - Exact transportation problem solution
  - Real-life constraints

- Analysis of update patterns from several IBM-hosted web sites
  - Grand Slam Tennis
    - Australian, French, US Opens; Wimbledon
  - Golf
    - Master’s, Ryder’s Cup
  - Olympic Games
    - Nagano, 1998; Sydney 2000
  - Awards
    - Tonys, Grammys

- Summary: Interupdate time distributions span wide range of behaviors

Related Work

- Cho & Garcia-Molina (2000)
  - Compare “uniform” and “proportional” allocation rules
  - Solved crawling frequency problem for Poisson updates

- Coffman, Liu, & Weber (2000) consider Poisson updates, and also include *access* times.
  - crawls to a page should be as “evenly spaced” as possible (to maximize average freshness)

- Talim, Liu, Nain, & Coffman (1999): optimize *number* of crawlers. (Indexing capacity v/s starvation of the indexing engine.)

Key Notation

Variable	Description
N	Number of web pages to be crawled
T	Web crawler scheduling interval length
R	Number of crawls allowed in interval $[0, T]$
C	Number of crawlers
S_k	Number of crawls by crawler k in time T

Table 1: Notation summary

Crawling Frequency Problem: Preliminaries

- Suppose we crawl web page i x_i times during scheduling interval T...
  - ...at times $0 \leq t_{i,1} < t_{i,2} < \ldots < t_{i,x_i} \leq T$
  - ...with $x_i \leq R$.
- Compute probability $p_i(t_{i,1}, \ldots, t_{i,x_i}, t)$ that search engine will have stale copy of web page i at time $t \in [0, T]$:
- Compute *time-average* staleness probability measure

$$a_i(t_{i,1}, \ldots, t_{i,x_i}) = \frac{1}{T} \int_0^T p_i(t_{i,1}, \ldots, t_{i,x_i}, t) dt. \tag{1}$$

- Set *staleness*

$$\mathcal{A}_i(x_i) = a_i(t^*_{i,1}, \ldots, t^*_{i,x_i}). \quad (2)$$

- Set
  - w_i to be *embarassment* weights
  - m_i to be minimum number of acceptable crawls for web page i
  - M_i to be Maximum number of acceptable crawls for web page i

Crawling Frequency Problem:

Minimize

$$\sum_{i=1}^{N} w_i \mathcal{A}_i(x_i) \qquad (3)$$

subject to the constraints

$$\sum_{i=1}^{N} x_i = R$$
$$x_i \in \{m_i, \ldots, M_i\}.$$

- Solution to *discrete, separable resource allocation problem (RAP)*:
  - $O(NR^2)$
- But $\mathcal{A}_i$ is *convex*
  - So much faster algorithms exist

Crawling Frequency Problem: Loose Ends

- Computing the weights w_i of the *embarassment level* metric for each web page i

- Computing the functional forms of p_i, a_i and $\mathcal{A}_i$ for each web page i, based on the update pattern distribution

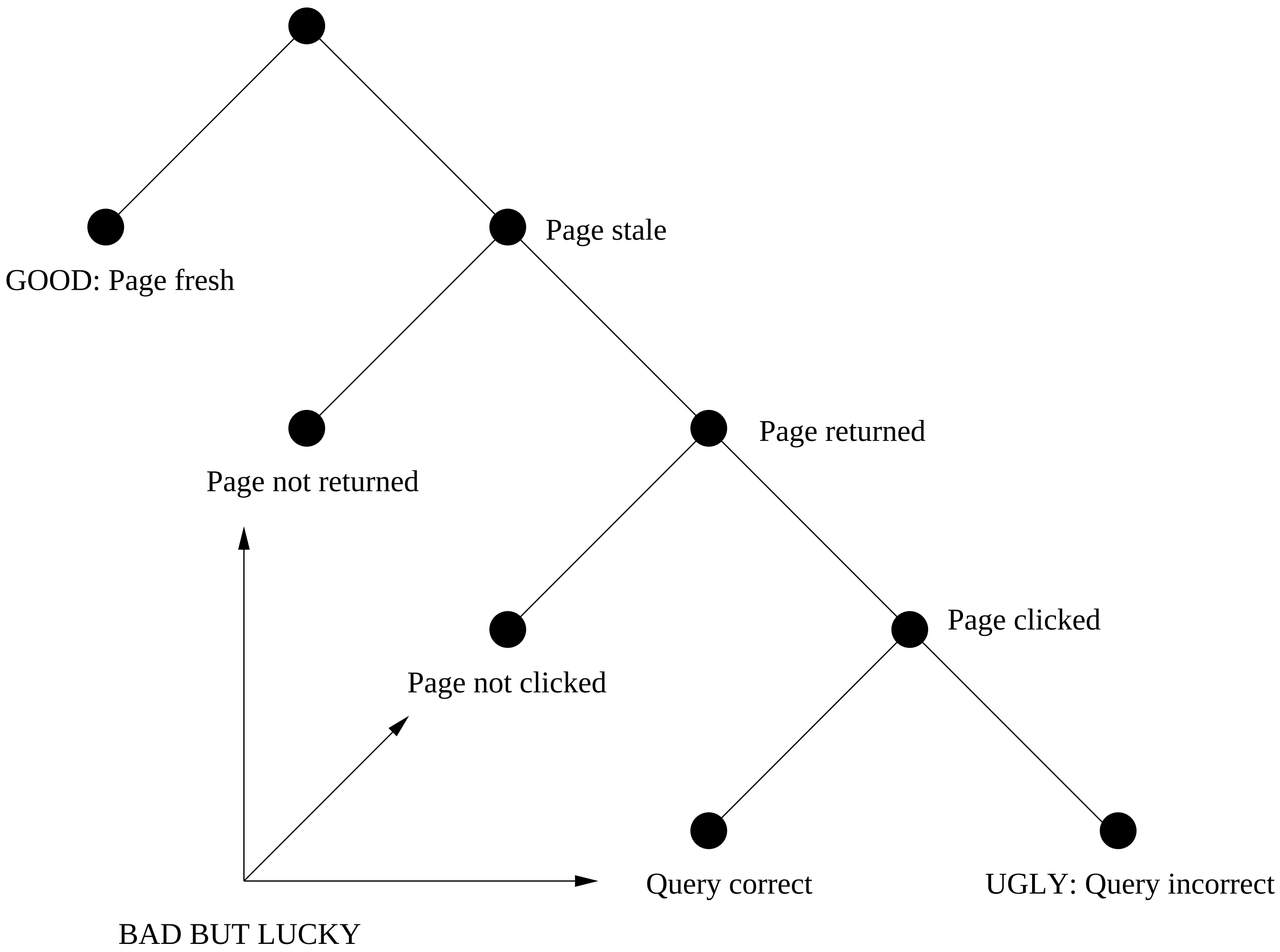
- Solving the resulting discrete, convex, separable resource allocation problem

Embarrassment Metric

- *Good*: Search engine has fresh copy of web page

- *Bad*: Stale...
  - ...but *lucky*
    - ∗ ♣ Web page not returned to client as result of query
    - ∗ ♣♣ Returned to client, but not clicked on by client
    - ∗ ♣♣♣ Returned to client, clicked on but correct with respect to query
  - ...and *unlucky*
    - ∗ Returned to client, clicked on...
      - · ♢♠♡ ...and incorrect with respect to query
      - · ♢♠♡ ...or gone
      - · Up to 14% of search engine links are typically broken

Computing the Weights w_i



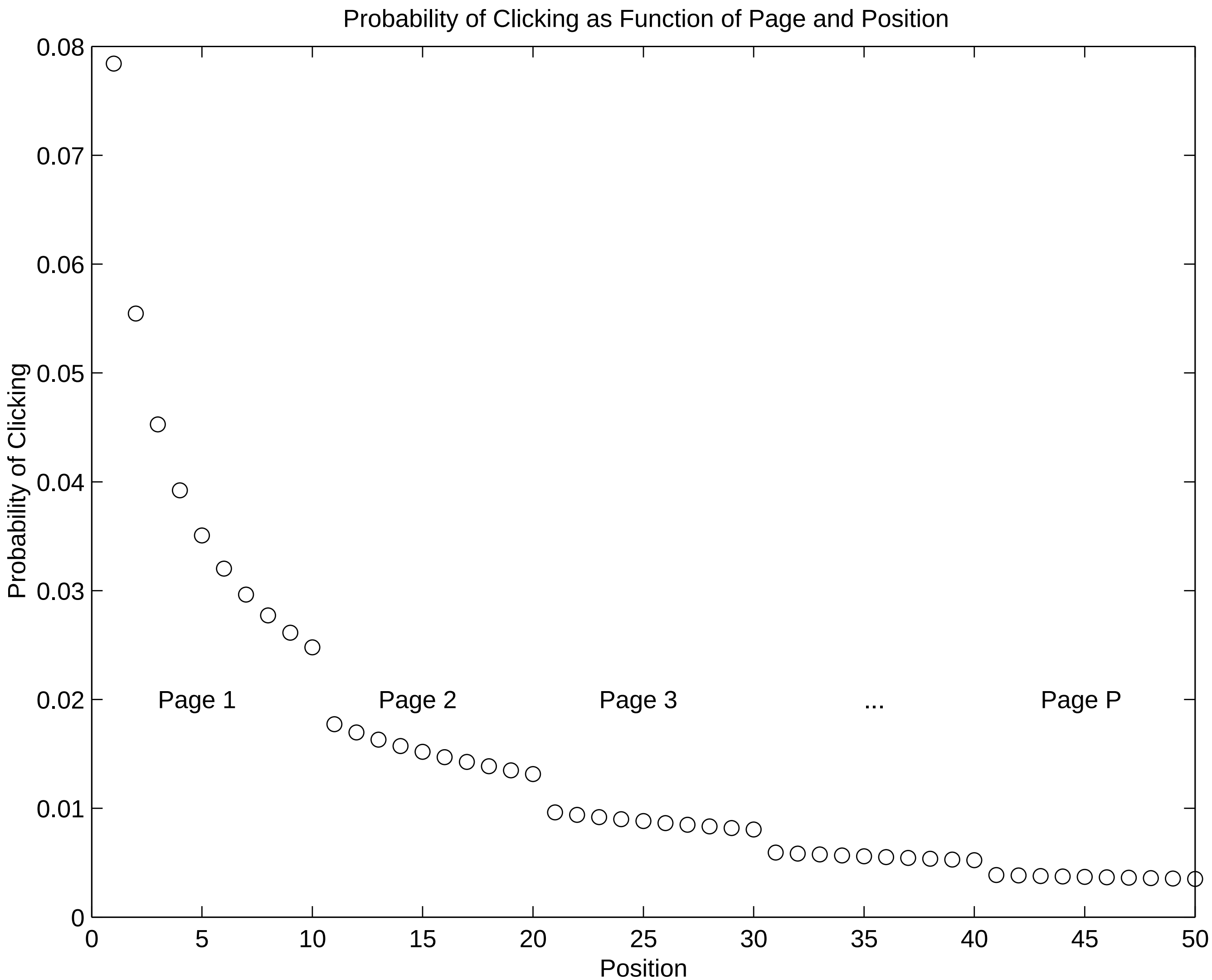
- Let $b_{i,j,k}$ denote the probability that the search engine will return page i in position j of query result page k

- Let $c_{j,k}$ denote the frequency that a client will click on a returned page in position j of query result page k

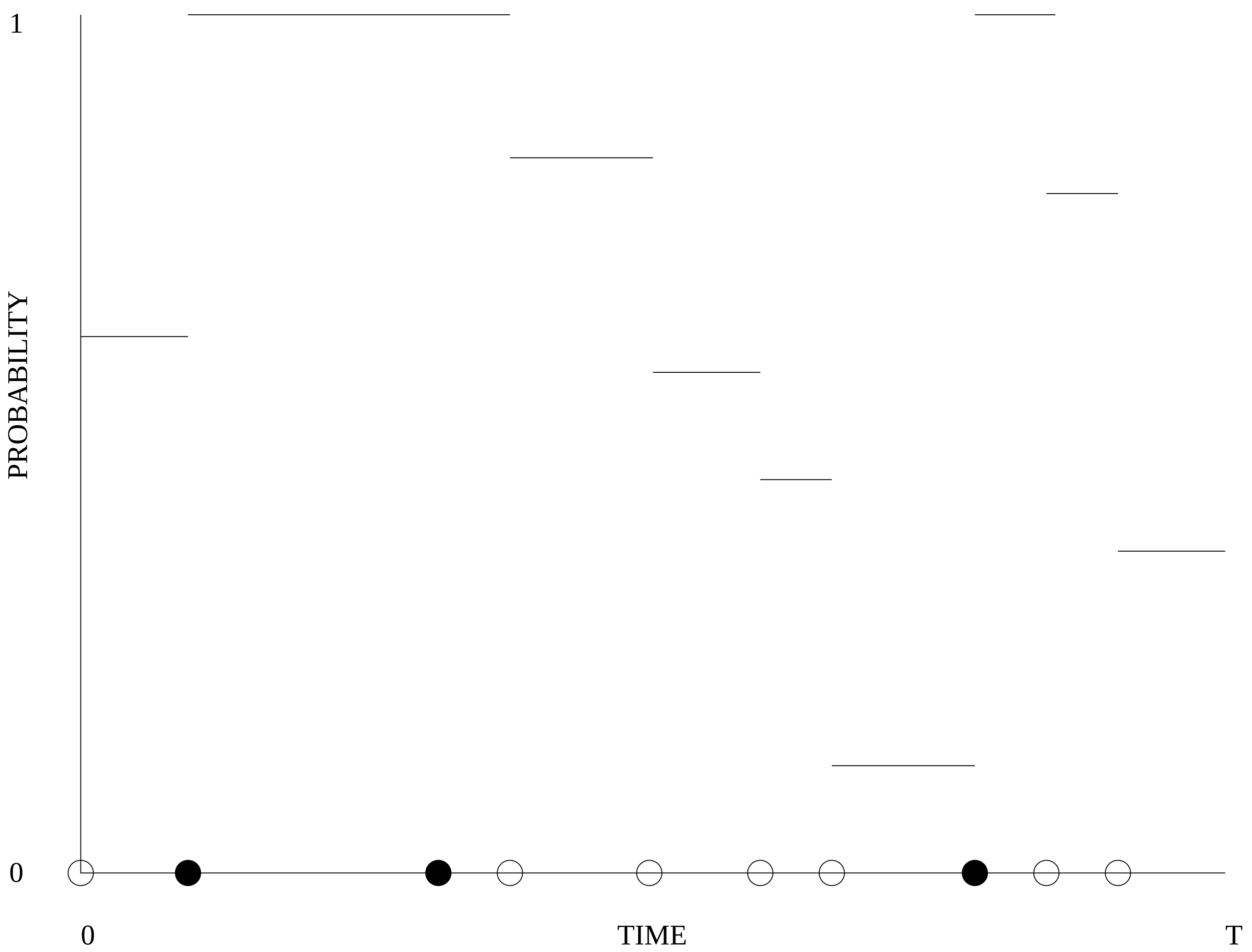
- Let d_i denote the probability that a query to a stale version of page i yields an incorrect response

- Then $w_i = d_i \sum_j \sum_k c_{j,k} b_{i,j,k}$

Probability of Clicking As Function of Position/Page



Probability Function for Quasi-Deterministic Web Pages



Computing the Functions $\bar{p}_i$, $\bar{a}_i$ and $\mathcal{A}_i$: Quasi-Deterministic Case

- P_i possible update times, $0 \leq s_{i,1} < s_{i,2} < \ldots < s_{i,P_i} \leq T$
- Update $s_{i,j}$ occurs with probability $q_{i,j}$
- $u_{i,j} \; = \; s_{i,j+1} - s_{i,j}$
- Can assume $x_i \leq P_i + 1$
- Define binary decision variables

$$y_{i,j} = \begin{cases} 1 & \text{if a crawl occurs at time } s_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- Define

$$J_{1,t} = max\{j|0 \leq j \leq x_i,\ s_{i,j} \leq t\}, \quad (5)$$

which is index of latest potential update time before t; and

$$J_{2,t} = max(0, \{j|0 \leq j \leq x_i,\ y_{i,j} = 1,\ s_{i,j} \leq t\}), \quad (6)$$

which is index of latest potential update time before t that will be crawled

- Then, the *freshness* probability function is:

$$\bar{p}(y_{i,0}, \ldots, y_{i,P_i}, t) = \prod_{j=J_{2,t}+1}^{J_{1,t}} (1 - q_{i,j}) \quad (7)$$

- The *average staleness* is

$$\bar{a}(y_{i,0}, \ldots, y_{i,P_i}) = \sum_{j=0}^{P} u_{i,j}(1 - \prod_{k=J_{i,j}+1}^{j} (1 - q_{i,j})) \tag{8}$$

To compute $\mathcal{A}_i$, minimize

$$\bar{a}(y_{i,0}, \ldots, y_{i,P_i}) \tag{9}$$

Subject to the constraints

$$y_{i,j} \in \{0, 1\} \tag{10}$$

And

$$\sum_{j=0}^{P} y_{i,j} = x_i \tag{11}$$

Quasi-Deterministic Case

- A simple dynamic program solves the problem optimally.

- *Greedy* algorithm does *exceptionally* well; is at least within e/(e-1) of optimal, possibly better. (Series of papers by Fisher, Nemhauser, Wolsey, Cornuejols and Conforti on maximizing submodular functions)

  – Greedy's $\mathcal{A}_i$ is also convex!

Computing the Functions p_i, a_i and $\mathcal{A}_i$: Poisson Case

- Probability of update occurring within t units of time: $1 - e^{-\lambda_i t}$
- Suppose we crawl x_i times at times $t_{i,1}, \ldots t_{i,x_i}$
- Define $\hat{t} = max\{t_{i,j} | 0 \leq j \leq x_i,\ t_{i,j} \leq t\}$
- Then $p_i(t_{i,1}, \ldots, t_{i,x_i}, t) = 1 - e^{-\lambda_i (t - \hat{t})}$
- And $a_i(t_{i,1}, \ldots, t_{i,x_i}) = 1 + \frac{1}{\lambda_i T} \sum_{j=0}^{x_i} \left(e^{-\lambda_i (t_{j+1} - t_j)} - 1\right)$
- So let $u_{i,j} = t_{i,j+1} - t_{i,j}$

Poisson Case

To find $\mathcal{A}_i$ minimize

$$1 + \frac{1}{\lambda_i T} \sum_{j=0}^{x_i} (e^{-\lambda_i u_{i,j}} - 1) \tag{12}$$

subject to the constraints

$$0 \leq u_{i,j} \leq T, \tag{13}$$

and

$$\sum_{j=0}^{x_i} u_{i,j} = T. \tag{14}$$

- A continuous, convex, separable resource allocation problem: solution is trivial.

- So $u^*_{i,j} = T/(x_i + 1)$

A Unified Framework

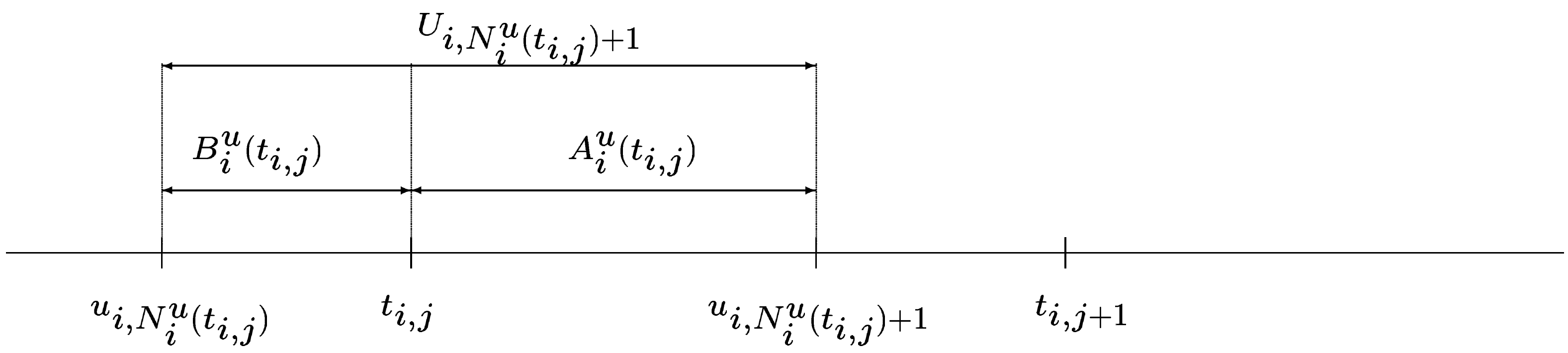


Figure 1: Example of a Marked Point Process

A Unified Framework

- From standard renewal-theoretic arguments, the time-average staleness can be computed as:

$$a_i(t_{i,1},\ldots,t_{i,x_i}) \quad = \quad \frac{1}{T}\sum_{j=0}^{x_i}\int_{t_{i,j}}^{t_{i,j+1}}\left(1 \,-\, \lambda_i\int_0^{\infty}\overline{G}_i(t-t_{i,j}+v)\,dv\right)dt.$$

Here $\overline{G}_i(\cdot)$ is the complementary c.d.f. of the inter-update time distribution.

NOTE: Summands are *separable*, *identical*! So optimal crawl times are evenly spaced.

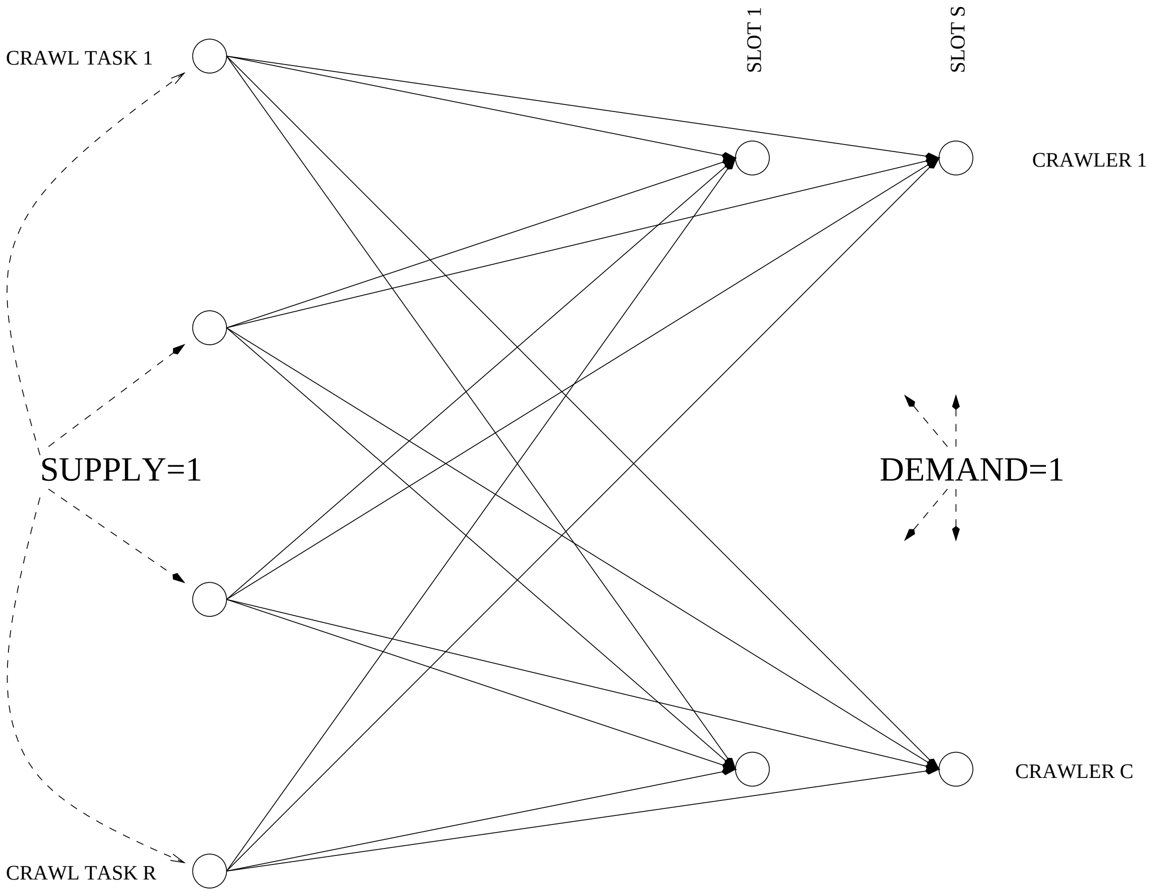
Solving the Discrete, Separable, Convex RAP

- Fox greedy algorithm
  - $O(N + R \log N)$
- Galil and Megiddo algorithm
  - $O(N(\log R)^2)$
- Frederickson and Johnson algorithm
  - $O(\max\{N, N \log(R/N)\})$
- Continuous relaxation / bracket and bisection algorithm also possible

Crawler Scheduling Problem

- C crawlers
  - Crawler k can handle S_k crawls in time T
- Time slots $T_{k,l}$
- Cost functions
  - For stochastic web pages $S(t) = |t - t^*_{i,j}|$
  - For quasi-deterministic web pages $S(t) = \begin{cases} \infty & \text{if } t < t^*_{i,j} \\ t - t_{i,j} & \text{otherwise} \end{cases}$
- Solvable as *transportation* problem

Transportation Problem Network



Transportation Problem Formulation

$$\text{Minimize } \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M} R_i(T_{jk}) f_{ijk} \qquad (15)$$

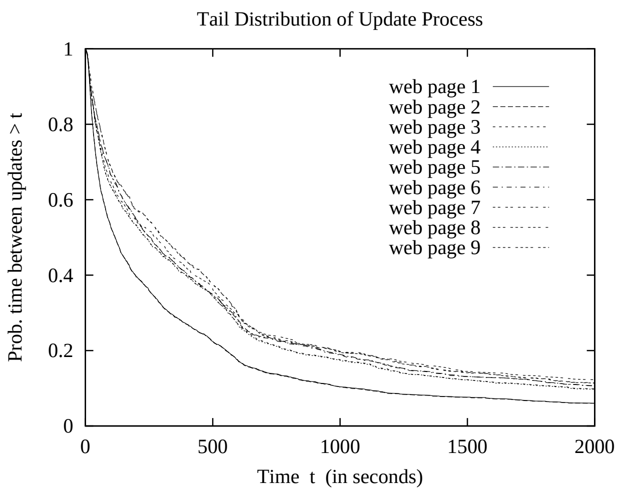
such that

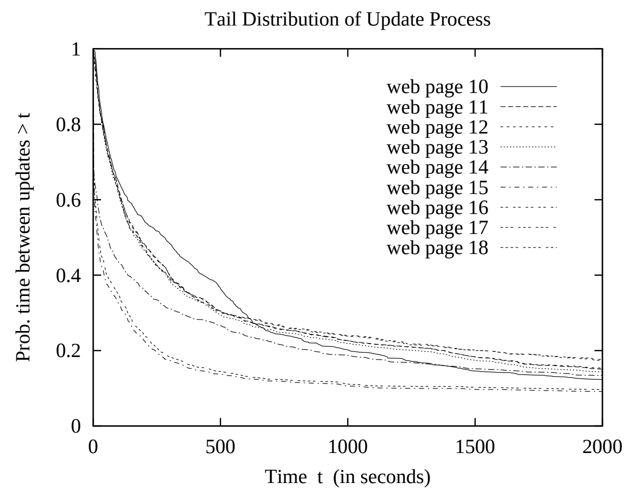
$$\sum_{i=1}^{M} f_{ijk} = 1 \; \forall \; 1 \leq j \leq N \text{ and } 1 \leq k \leq M \qquad (16)$$

and

$$\sum_{j=1}^{N}\sum_{k=1}^{M} f_{ijk} = 1 \; \forall \; 1 \leq i \leq M \qquad (17)$$

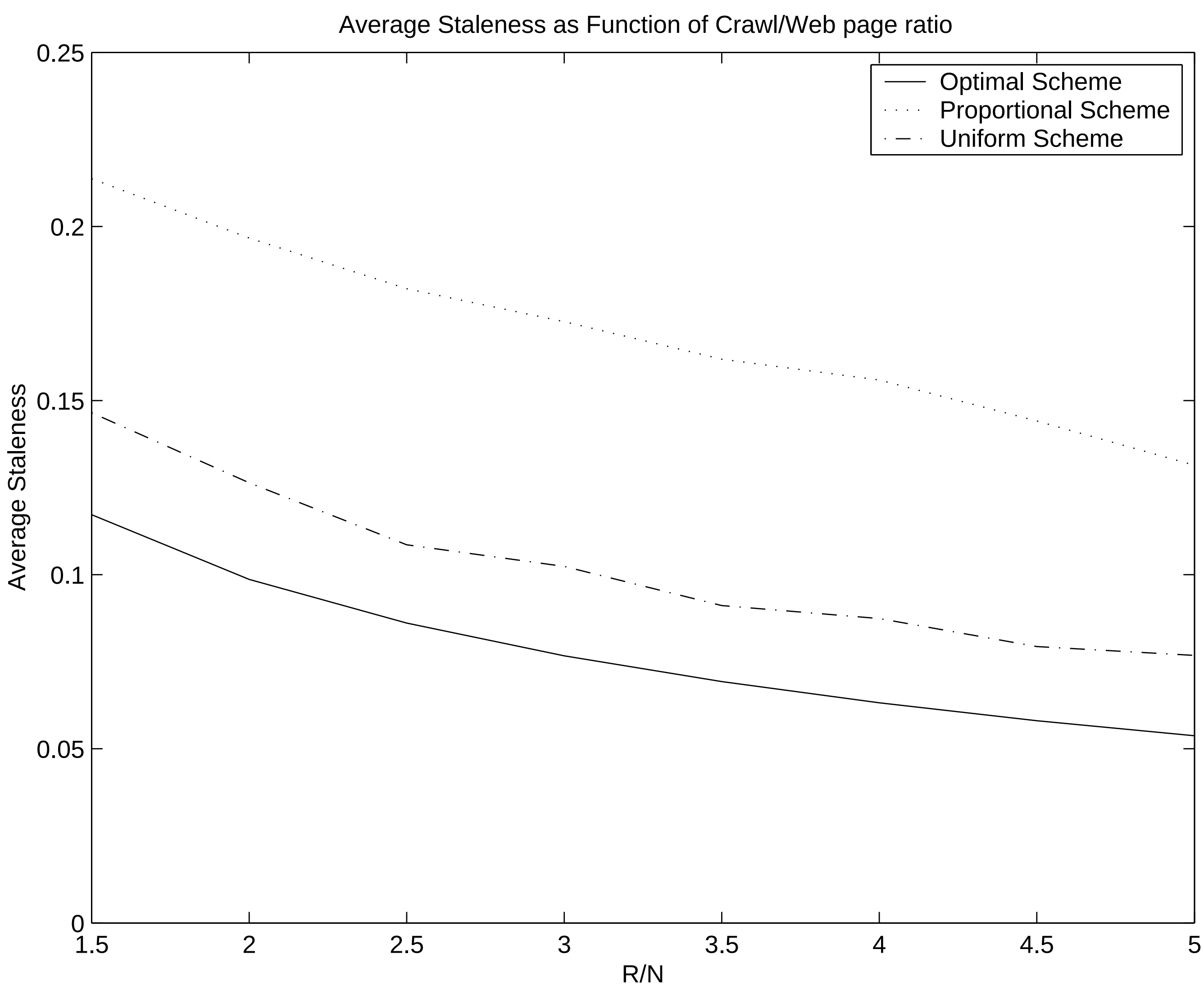
Tail Distributions of Update Processes



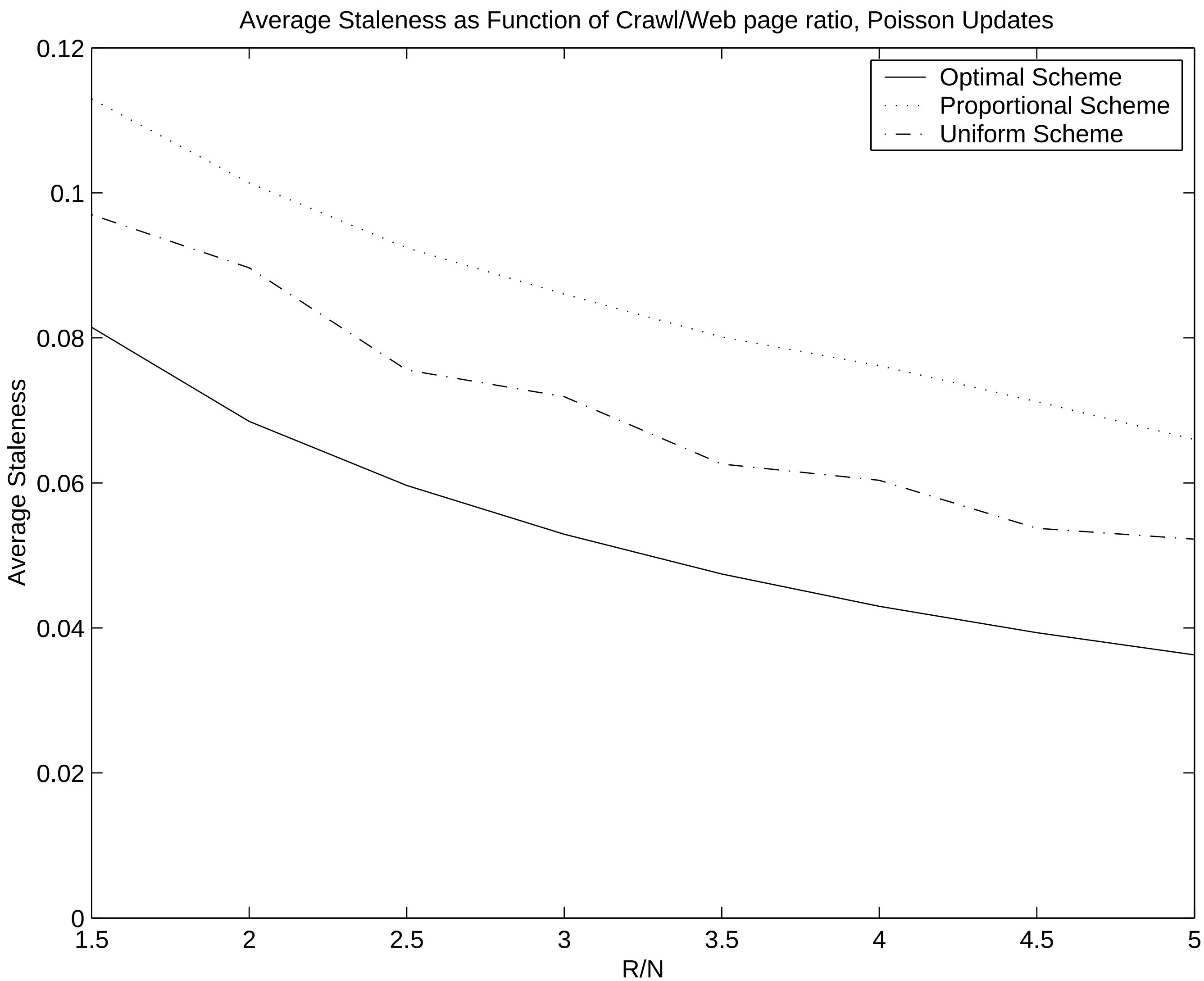


- Real web logs
- Can vary across many distribution types
  - Not typically Poisson
  - Often Weibull
  - Sometimes Pareto
  - Quasi-deterministic examples also exist

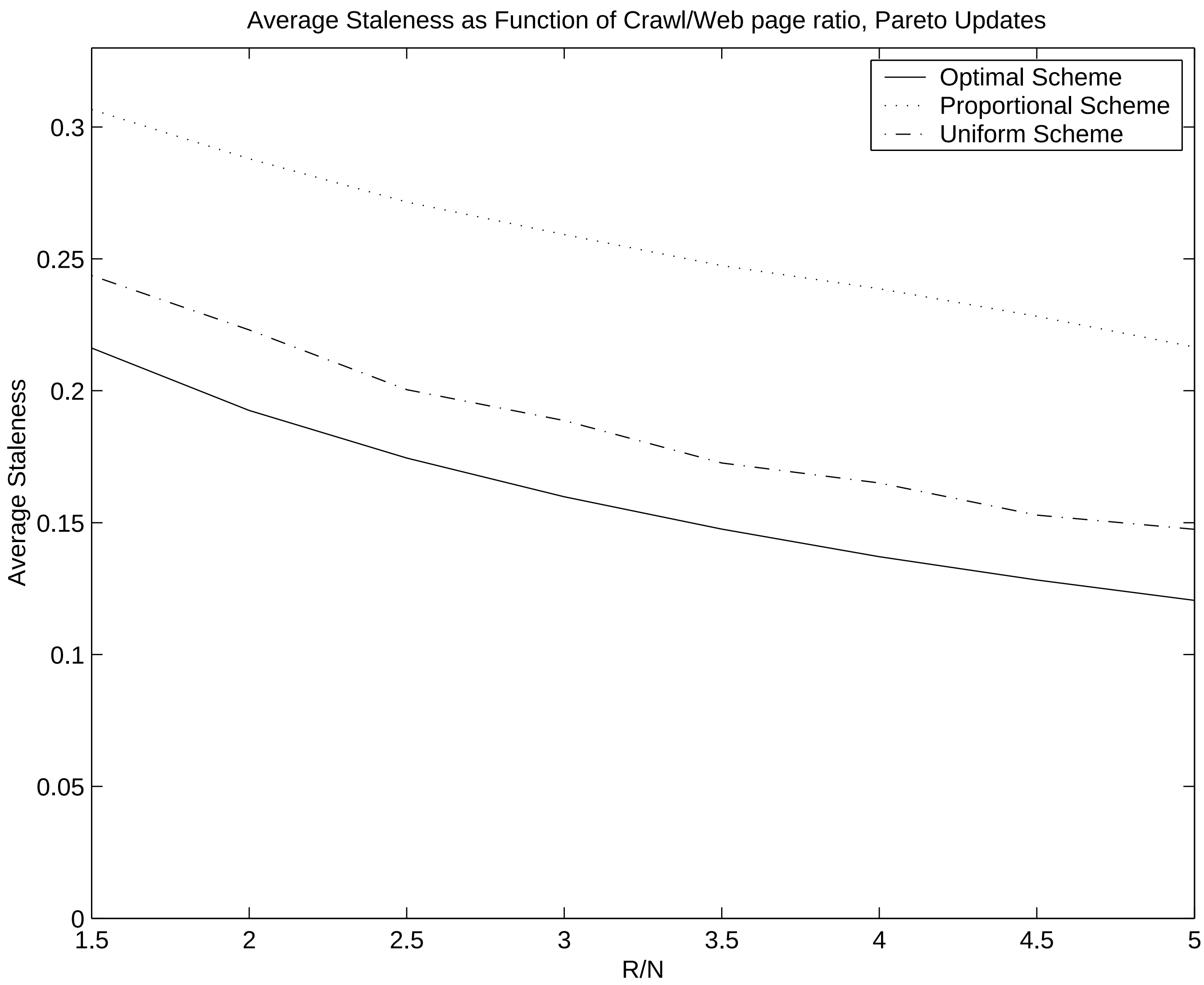
Average Staleness Metric Examples: Mixed



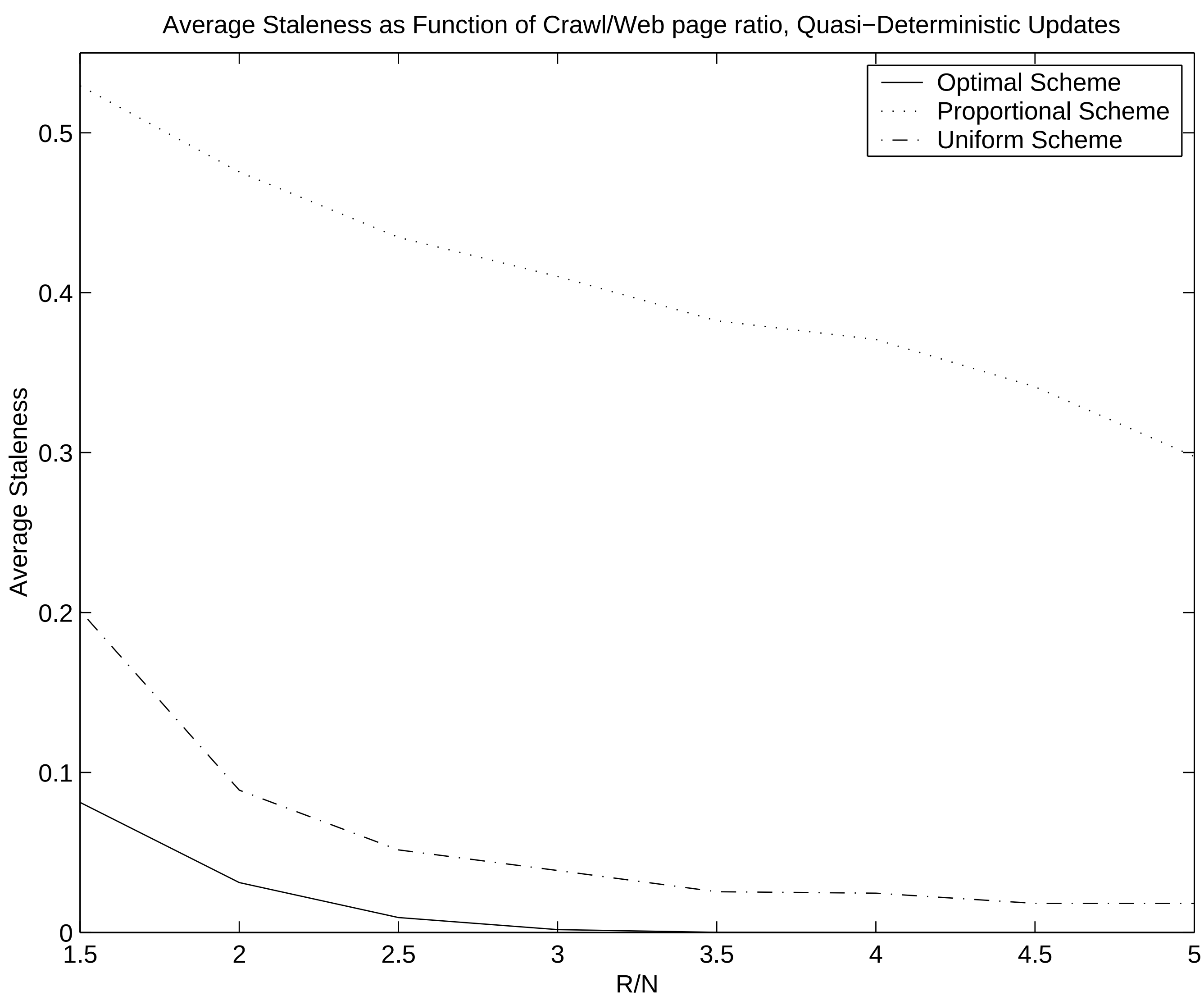
Average Staleness Metric Examples: Poisson



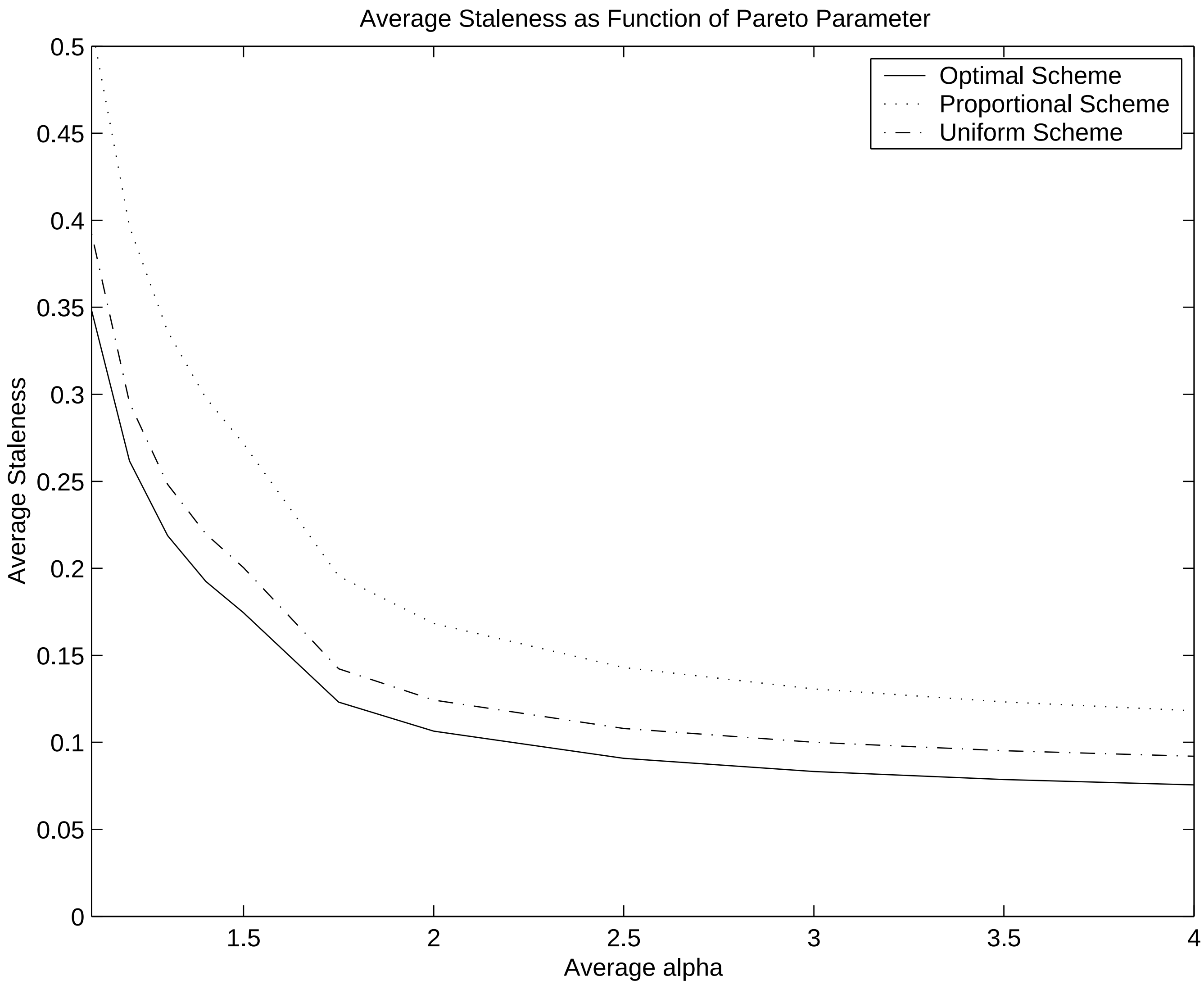
Average Staleness Metric Examples: Pareto



Average Staleness Metric Examples: Quasi-Deterministic



Pareto Example



Scheduling Problem Example

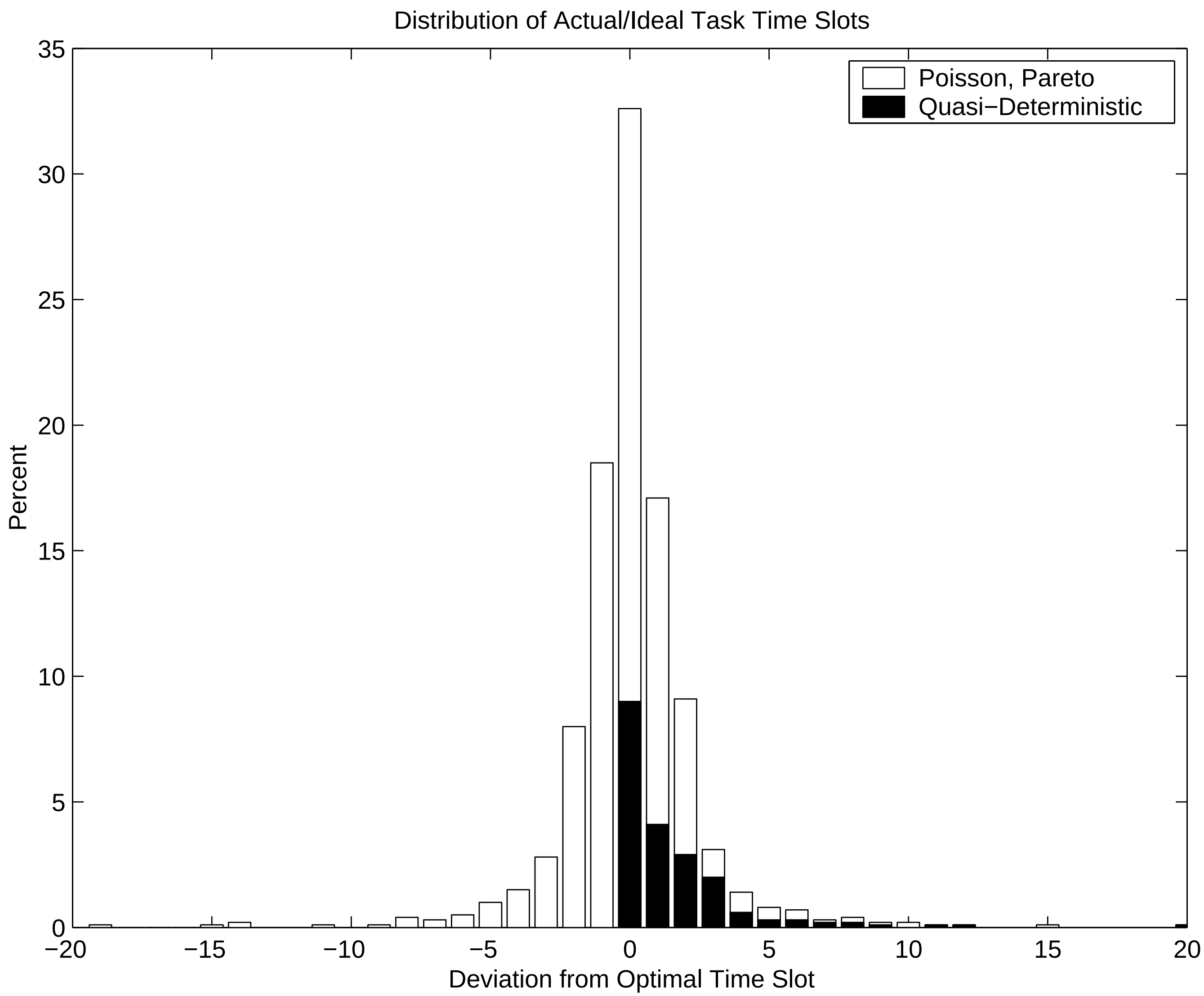


Figure 2: Transportation Problem Solution

Conclusions

- New formulation and solution for important search engine crawler optimization problem
  - Crawling frequency problem
    - Embarassment metric
    - General update distributions
    - Extraordinarily fast RAP solution
  - Crawler scheduling problem
    - Transportation problem formulation
- Study of real web log data
  - Shows many distribution patterns