大規模 Web アーカイブ更新クローラにおけるスケジューリング手法の評価

田村　孝之[†,††a)]　　喜連川　優[††b)]

Evaluation of Scheduling Methods of an Incremental Crawler for Large Scale Web Archives

Takayuki TAMURA[†,††a)] and Masaru KITSUREGAWA[††b)]

あらまし　Web 情報に基づく社会動向の分析には，刻々と変化する Web を高い網羅性と高い時間分解能でとらえた大規模 Web アーカイブが不可欠であり，各 Web ページの更新頻度を推定して適応的に再アクセスを行う更新クローラの実現が求められる．本論文では Web ページの更新有無の履歴に基づいて次回更新タイミングを推定する Web ページ再アクセススケジューリング手法を提案し，実データを用いてその有効性の評価を行う．提案手法は Poisson 過程における最ゆう推定を簡略化し，Web ページごとに保持する状態変数を大幅に削減することにより，大規模 Web ページ集合への適用を可能にした．また，高い時間分解能をもつ評価用データとしてフィード URL 集合をほぼ毎日アクセスした結果を用いたシミュレーションにより，提案手法が Web ページの更新間隔に適応しつつ更新の 90%程度をとらえられることが明らかになった．

キーワード　Web アーカイブ，更新クローラ，更新間隔推定，適応的スケジューリング，性能評価

1. まえがき

刻々と変化する膨大な Web 情報は実社会の鏡像であり，その大域的なグラフ構造や時間変化を分析することで実社会に関する高度な知識が抽出できると考えられる．筆者らはこうした Web 分析の基盤として，日本語 Web ページを中心とする大規模 Web アーカイブの構築に取り組んでいる [1]〜[3].

Web の大域的構造と時間変化の分析を主眼とする Web アーカイブに要求される網羅性と時間分解能の両立を図るには，1 回の動作で各 Web ページを 1 回ずつアクセスする一括クローラ (batch crawler) ではなく，連続的に動作し続け，Web ページごとに独立したタイミングで再アクセスを行う更新クローラ (incremental crawler) が不可欠である．更新クローラは過去のクローリング履歴をデータベース化し，各 Web ページの更新頻度に適応したスケジューリングを行うことで，低更新頻度 Web ページのアクセスに割かれていたリソースを高更新頻度 Web ページのアクセスに振り向け，最短 1 日程度の高い時間分解能を実現する．

本論文では，更新クローラ実現の要となる Web ページ再アクセススケジューリング手法を提案し，大規模な実データを用いた性能評価結果について述べる．提案手法は Web ページの平均更新間隔推定に基づくが，推定精度よりも動作の単純さを重視し，大幅な近似を行った．大規模 Web クローリングへの適用においては，Web ページの取得ごとに発生するスケジューリング処理の負荷や Web ページごとに維持する状態量の容量が問題となるためである．また，従来の関連研究において重視されてきたのは，クロール結果と最新 Web 状態の乖離や更新非検出アクセスの発生を低減することであったが，本論文では Web アーカイブ構築を目的とするため，Web ページの各バージョンをどれだけ網羅できたかに主眼を置く．そのため，評価データとしては，1 年以上にわたってほぼ毎日アクセ

† 三菱電機株式会社情報技術総合研究所，鎌倉市
Information Technology R&D Center, Mitsubishi Electric Corporation, 5–1–1 Ofuna, Kamakura-shi, 247–8501 Japan
†† 東京大学生産技術研究所，東京都
Institute of Industrial Science, The University of Tokyo, 4–6–1 Komaba, Meguro-ku, Tokyo, 153–8505 Japan
a) E-mail: ttamura@acm.org
b) E-mail: kitsure@tkl.iis.u-tokyo.ac.jp

スし，更新（発行）タイミングが正確に判明しているRSS 等のフィード更新ログを用いた．

以下，**2.** で関連研究について述べ，**3.** で本論文が対象とする更新クローラのモデルについて説明する．**4.** では提案する Web ページ再アクセススケジューリングアルゴリズムを比較対象の逐次制御手法とともに述べる．**5.** ではフィード更新ログを用いたシミュレーションにより提案手法の性能評価を行い，**6.** で全体のまとめを行う．

2. 関連研究

日本を含め各国で国立図書館を中心に Web アーカイビングプロジェクトが組織されている [4]〜[6]．これらのプロジェクトは文化的な背景から Web 上のディジタル文書を保全することを目的としており，規模よりもコンテンツの再現性（スクリプトやプラグインの動作まで含む）が重視される傾向にある．米国の Internet Archive [7] は古くから網羅的な収集に取り組んでいるが，1〜2 か月ごとに外部から提供されるデータが主体となっている．Internet Archive 独自のオープンソースクローラの開発も進められているが，大規模化への取組みは始まったばかりである [8]．また，更新クローリングのサポートが検討されたものの，断念されたようである [9], [10]．

Web ページ更新傾向の観測に基づく研究としては文献 [11], [12] があり，Web ページ更新は単純な Poisson 過程としてある程度モデル化できることが確認されている．Cho ら [13] は一定間隔で Web ページをアクセスした際に，検出された更新の有無から Poisson 過程のパラメータとして Web ページの更新頻度を推定する手法について述べている．本論文の提案手法でも同様のモデルに基づいて更新頻度の推定を行うが，大規模クローリングへの適用を考慮し，実際の Web ページ収集に伴う不規則なアクセス間隔を前提としている．一方，1 日未満などの短い期間については，更新頻度が定数ではなく時間とともに変化する非斉次 Poisson モデルを用いる必要があることが指摘されている [14]．

WebFountain クローラ [15] は特定の Web ページ更新モデルに依存せず，各ページの更新確率やページ収集後の経過時間などを変数とする非線形連立方程式を一定期間ごとに解いて収集対象 Web ページを選択する．実験で用いられたコスト設定では，更新確率が高いページの収集はクローリング終了間際に行うという挙動を示している．同様に，Cho ら [12], [16] は Web ページの更新頻度に基づいてアクセスの優先度を決定する方法を論じており，特に，更新頻度が非常に高い Web ページは取得した内容が最新である期間も短いため，他の Web ページの収集を優先した方がスナップショット全体の新鮮度が向上するという結論を得ている．しかし，これらの方針は最新情報の維持を目的とすることから導き出されるものであり，過去の情報も現在の情報と同等の重要性をもつ Web アーカイブ用途においては必ずしも妥当でない．更新頻度が高い Web ページは Web アーカイブにおいても多くのバージョンが保存されていると考える方がむしろ自然であろう．Web アーカイブ構築のためのクローリングにおいては新鮮度 (freshness) や経過時間 (age) ではなく，バージョンの網羅率に着目した評価が重要と考えられる．

一方，大量の Web ページを扱うことの困難さを回避するために，Web サイトや Web ページ群を単位として更新の有無を推定する手法も提案されている [17], [18]．これらの手法では 1 か月あるいは 1 週間などの比較的長い間隔で一部の Web ページをサンプリングし，更新が検出された場合に残りの Web ページを一括してクロールするという戦略を用いる．しかし，更新されていない Web ページの収集コスト (false positive) や更新された Web ページの収集漏れ (false negative) の問題が本質的に伴うため，大規模な Web ページ集合を高い時間分解能で収集するのは難しい．

Web サイト側が RSS 等のフィードを提供している場合には，更新された Web ページが列挙されているので効率的に Web ページの更新をとらえることができる [14]．また，Web サイト側がクローラへのヒント情報として更新頻度や優先順位とともに Web ページ URL を列挙する sitemap プロトコルも提案されている [19]．これらの協調的な仕組みはすべての Web サイトや Web ページを網羅したものではなく，単独で大規模 Web アーカイブの構築に適用することはできないが，本論文の提案手法を補完するものとして更新クローラの改善に有効と考えられる．なお，**5.** の性能評価においては，フィード更新ログを Web ページ更新時系列の生成に用いているが，更新頻度等のフィード固有の情報は利用していない．

3. 更新クローラの動作モデル

図 1 に本論文で扱う大規模 Web アーカイブ構築用更新クローラの構成を示す．筆者らが実装した更新ク

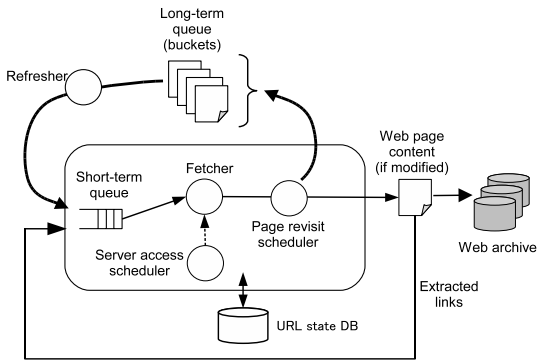

図 1　更新クローラの構成
Fig. 1　Architecture of incremental crawler.

ローラはクラスタ上で並列動作するが，既知 Web ページの再アクセスは各ノードが独立に行うため他ノードとの通信を意識する必要はない．

更新クローラは以下のように動作する．

（1） Server access scheduler は Web サーバごとに一定の時間間隔で short-term queue から URL を取得し，fetcher を起動する．Short-term queue は論理的には Web サーバごとに独立したプライオリティキューの集合とみなすことができ，Web サーバごとに先頭 URL の取得や指定 URL の削除及び挿入が可能である．

（2） Fetcher は受け取った URL から HTTP 要求メッセージを作成し，Web サーバに送信する．その際，Web サーバがサポートしていれば conditional GET メソッド [20] を用い，更新のないコンテンツの再ダウンロードを避けるようにする．そのため，当該 URL を前回ダウンロードした際に HTTP 応答ヘッダから抽出した Last-Modified 及び ETag フィールド情報を URL state DB から取得して HTTP 要求ヘッダに設定する．

（3） Fetcher は Web サーバから応答を受け取ると，応答コードが 304 (Not Modified) であるか，あるいはコンテンツのハッシュ値が URL state DB から取得した前回ダウンロード時のハッシュ値と等しい場合に更新がなかったと判定する．Web ページが更新されていた場合はそのコンテンツを Web アーカイブに格納し，更に，コンテンツが HTML であればリンク抽出処理を起動する．

（4） Fetcher は HTTP 応答ヘッダから抽出した情報やコンテンツハッシュ値，現在時刻等を用い，URL state DB の当該 URL に関するレコードを更新する．

（5） Page revisit scheduler は URL state DB の情報に基づき，当該 URL の次回アクセス時刻を決定する．更に，URL を short-term queue から取り除き，long-term queue に挿入する．Long-term queue は Web ページ再アクセススケジューリングの時間分解能単位の時刻に対応するバケットファイル群から構成されており，更新操作は各バケットへの URL の追加のみを許す．既知 URL の大半は long-term queue に存在するため，厳密な順序管理を避けることで挿入コストの低減を図っている．

（6） Refresher は時間分解能ごとに起動され，long-term queue の最も早い時刻に対応するバケットファイルを取り出し，その中の各 URL を short-term queue に挿入する．

以上のように本更新クローラでは，各 URL のアクセス予定時刻を明示的に扱う．関連研究には更新頻度や更新確率に基づいてアクセス対象 Web ページを相対的に選出するものが多いが，上記 page revisit scheduler は Web ページの更新間隔に基づいて次回アクセス時刻を決定し，そのタイムスロットを予約するという動作を行う．更新間隔を扱う方がクローラの動作を直観的に把握しやすいと思われるが，基本的に両者は双対であり，本質的な優劣はない．どちらの概念を陽に扱うかはクローラの構造にも反映されており，例えば WebFountain クローラでは URL を更新確率ごとのバケットに分割して管理している [15]．

なお，（3）の更新有無判定はコンテンツの単純な一致・不一致に基づいて行っているため，主要コンテンツが不変でアクセスカウンタや新着ニュースへのナビゲーションリンク等が変化しているにすぎない場合も更新と判断することになる．本質的でない更新を無視する技法としては，w-shingling や simhash などのテキスト類似判定手法 [21]〜[23] や，HTML の DOM 構造から主要コンテンツを特定する手法 [24], [25] 等が提案されている．十分な精度を得るための評価やチューニングを行った上で，これらの技法を更新クローラに組み込むことを検討中である．

4. Web ページ再アクセススケジューリング手法

4.1 更新間隔の最ゆう推定に基づく手法

4.1.1 不規則アクセスに対する更新間隔推定の近似

ここでは，更新クローラの page revisit scheduler

によるスケジューリング処理の基礎として，Web ページを不規則にアクセスした際の更新有無の履歴から当該ページの平均更新間隔を推定する手法について述べる．Cho ら [13] は Web ページの更新が一定の平均更新頻度 λ（平均更新間隔の逆数）をもつ Poisson 過程に従うと仮定し，ある Web ページを一定間隔でアクセスした際の更新検出結果から λ を推定する方法について包括的に議論している．しかし，大規模 Web アーカイブ構築においては大量の Web ページを扱うため，更新間隔推定のためにサンプリングアクセスを高頻度に繰り返すのは不適切であり，推定した更新間隔を直ちに次回アクセス間隔に反映させて不必要なアクセスを避ける必要がある．

ここで図 2 のとおり不規則アクセスによるページ更新検出を表す変数を導入する [13]．すなわち，あるページに対して初回アクセス後に n 回アクセスした際に m 回 $(0 < m < n)$ の更新を検出したものとし，第 i 番目 $(1 \leq i \leq m)$ の更新を検出した際のアクセス間隔を t_{ci}，更新が検出されなかったアクセス間隔の第 j 番目 $(1 \leq j \leq n-m)$ のものを t_{uj} とする．

Poisson 過程では t_{ci} の期間に 1 回以上の更新が起こる確率は $1 - \exp(-\lambda t_{ci})$，$t_{uj}$ の期間に更新が起こらない確率は $\exp(-\lambda t_{uj})$ であるから，実際にある更新パターンが観測される確率は式 (1) で与えられる．

$$\prod_{i=1}^{m}(1 - \exp(-\lambda t_{ci})) \prod_{j=1}^{n-m} \exp(-\lambda t_{uj}) \qquad (1)$$

λ の推定値は最ゆう法により，式 (1) を最大化する λ として得ることができる．式 (1) の対数をとったものを λ で微分して 0 に等しいとおくことにより，λ が満たすべき条件は以下の式 (2) となる．

$$\sum_{i=1}^{m} \frac{t_{ci}}{\exp(\lambda t_{ci}) - 1} = \sum_{j=1}^{n-m} t_{uj} \qquad (2)$$

この式は過去のアクセス間隔の履歴をすべて含んでおり，Web ページごとに URL state DB に保持する状態量が膨大になってしまう．また，保持する履歴の範囲を有限ウィンドウ内に限定したとしても，式 (2) から λ の値を求めるには数値解法が必要となる．そこで式 (2) を近似し，より単純な推定方法を導くことにする．

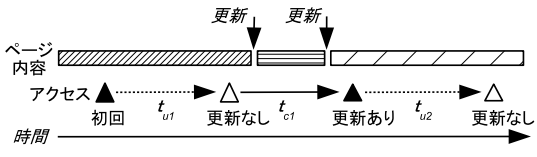

図 2　不規則アクセス間隔によるページ更新検出
Fig. 2　Page change detection with irregular access intervals.

まず，各 t_{ci} がすべて等しい場合，すなわち $t_{ci} = t_c$ $(1 \leq i \leq m)$ の場合について式 (2) を解く．この場合，全アクセス間隔の総和

$$T = \sum_{i=1}^{m} t_{ci} + \sum_{j=1}^{n-m} t_{uj}$$

と，更新が検出されなかったアクセス間隔の総和

$$U = \sum_{j=1}^{n-m} t_{uj}$$

を用いると，λ の推定値の逆数は式 (3) で与えられる．

$$\hat{\lambda}^{-1} = \frac{t_c}{\log \dfrac{T}{U}} \quad (0 < U < T) \qquad (3)$$

すべての t_{ci} が等しくない場合については，式 (3) の t_c を $\{t_{ci}\}$ の代表値 $\bar{t_c}$ で置き換えた式 (4) により λ の逆数を近似する．

$$\hat{\lambda}^{-1} \simeq \frac{\bar{t_c}}{\log \dfrac{T}{U}} \quad (0 < U < T) \qquad (4)$$

$\bar{t_c}$ としては例えば t_{ci} の最小値 $t_{c\min}$ や平均値 $t_{c\,\mathrm{avg}}$ を用いることが考えられる．最小値 $t_{c\min}$ を用いると真の推定値より小さな値が得られることになり，不要なアクセスの増加と引き替えに更新を取りこぼすリスクを減少させる効果がある．

式 (4) においては個々のアクセス間隔の履歴は不要であり，初回アクセスからの経過時間 T，更新が検出されなかったアクセス間隔を累積した U，更新を検出したアクセス間隔の最小値 $t_{c\min}$ や平均値 $t_{c\,\mathrm{avg}}$ $(= (T-U)/m)$ から次回アクセスのタイミングを決定することができる．このように Web ページごとの状態変数を大幅に削減することにより，大規模な Web ページ集合に対して更新間隔推定を行うことが可能となる．

4.1.2　次回アクセス間隔の決定

式 (4) を用いることができるのは $0 < \frac{U}{T} < 1$ の範囲に限られるが，実際のクローリングにおいては，全く更新が検出されない場合 $(m = 0,\ \frac{U}{T} = 1)$ やすべて

のアクセスで更新が検出される場合 ($m = n$, $\frac{U}{T} = 0$) も発生し得る．したがって次回アクセス間隔の決定にはこれらの場合も考慮に入れる必要がある．

式 (4) における $\bar{t_c}$ の乗数 $\mu = \left(\log \frac{T}{U}\right)^{-1}$ は $\frac{U}{T}$ の単調増加関数であり，$\mu \to 0$ ($\frac{U}{T} \to +0$)，$\mu \to \infty$ ($\frac{U}{T} \to 1{-}0$) となる．μ は Web ページの安定度 $\frac{U}{T}$ に応じてアクセス間隔を調整する効果をもつが，アクセス間隔の極端な変化を避けるため，下限 μ_l 及び上限 μ_h ($0 < \mu_l < \mu_h$) を設け，$\mu < \mu_l$ すなわち $\frac{U}{T} < e^{-\frac{1}{\mu_l}}$ となる範囲では $\mu = \mu_l$ とし，$\mu > \mu_h$ すなわち $\frac{U}{T} > e^{-\frac{1}{\mu_h}}$ となる範囲では $\mu = \mu_h$ とする．$\frac{U}{T} = 0$ の場合は $\frac{U}{T} < e^{-\frac{1}{\mu_l}}$ に含めることができる．一方，$\frac{U}{T} = 1$ の場合は Web ページが全く更新されず，$\bar{t_c}$ 自体が定義できないため，直前のアクセス間隔を基準に μ_h を乗ずることにする．

また，Poisson 過程においては平均よりも小さな間隔での更新の発生頻度が高いことを考慮し，式 (4) の平均更新間隔推定値に対し，$0 < \alpha \leq 1$ なる定数 α を乗じた値をアクセス間隔として用いることにする．

以上を総合し，第 $i-1$ 回アクセスから第 i 回アクセスまでの時間間隔 τ_i ($i > 2$) を式 (5) に基づいて決定する．

$$\tau_i = \begin{cases} \alpha\mu_l \cdot \bar{t_c} & (0 \leq \frac{U}{T} < e^{-\frac{1}{\mu_l}}) \\ \alpha \left(\log \frac{T}{U}\right)^{-1} \bar{t_c} & (e^{-\frac{1}{\mu_l}} \leq \frac{U}{T} \leq e^{-\frac{1}{\mu_h}}) \\ \alpha\mu_h \cdot \bar{t_c} & (e^{-\frac{1}{\mu_h}} < \frac{U}{T} < 1) \\ \mu_h \cdot \tau_{i-1} & (\frac{U}{T} = 1) \end{cases} \tag{5}$$

なお，初回アクセスから第 2 回アクセスまでの時間間隔 τ_2 には一定値を用いる．

4.1.3 Last-Modified 情報の利用

Web サーバによっては HTTP 応答ヘッダに Web ページが最後に更新された時刻を表す Last-Modified 情報を付与することがある．Last-Modified 情報は Web ページが更新された真の時刻と考えられ，より精度の高い更新間隔推定に利用できる．

図 3 は更新を検出したアクセスにおいて，Last-Modified 情報が利用可能であった場合の変数設定を表している．すなわち，前回アクセスから Last-Modified 情報の時刻までを更新検出アクセス間隔 t_{ci} とし，Last-Modified 情報の時刻から最終アクセスまでを更新非検出アクセス間隔 t_{uj} とする．これに対して，Last-Modified 情報が利用できない場合は $t_{ci} + t_{uj}$ が更新検出アクセス間隔とみなされ，U には寄与しない．

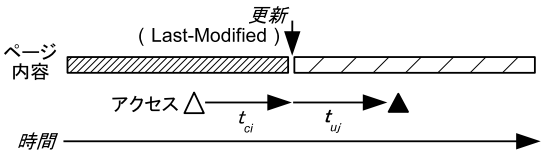

図 3 Last-Modified 情報が利用可能な場合の変数設定
Fig. 3 Variable settings under existence of Last-Modified information.

4.2 増加減少制御に基づく手法

更新間隔の最ゆう推定に基づく手法との比較対象として，Web ページの更新有無に基づき，直前のアクセス間隔を増加または減少させて次回のアクセス間隔とする制御手法を考える．増加減少制御では Web ページの状態変数として，直前のアクセス間隔（あるいは前回及び前々回のアクセス時刻）を保持するだけでよく，実現は容易である．

具体的には，TCP のふくそう回避等にも用いられている additive increase/multiplicative decrease (AIMD) アルゴリズムを採用する [26]．AIMD の動作は以下のとおりである．

$$\tau_i = \begin{cases} \tau_{i-1} + d & (\text{更新なし}) \\ \tau_{i-1} \times r & (\text{更新あり}) \end{cases} \tag{6}$$

ただし，定数 d，r は $d > 0$，$0 < r < 1$ とする．文献 [27] では，観測事象の発生頻度に適応してセンサを動作させ，エネルギー効率を改善するために AIMD が適用されている．

なお，AIMD では常にアクセス間隔が変化するが，更新非検出アクセスが連続した場合にアクセス間隔を増加させ，それまでは前回と同じアクセス間隔を用いる変形手法も考えられる．

5. 性能評価

5.1 フィード更新ログを用いたシミュレーション評価

更新クローラの実動作に対する評価では，小アクセス間隔領域において更新の取りこぼしの有無を確認することができないため，厳密な議論が難しい．そこで，高頻度に固定 URL 集合をアクセスした結果を用いてシミュレーション実験を行った．

有効なシミュレーションを行うためには，入力データの時間分解能の高さと対象 URL の多様さが求められる．しかし，大量の Web サーバに高頻度でアクセス

し続けることはマナー上好ましくなく，従来の研究においてはサイトを少数に限定したり，1 週間から 1 か月程度の長いアクセス間隔が用いられていた [17], [18]. Ntoulas ら [28] が 1 年間にわたって約 400 万 URL を 1 週間ごとにアクセスし，各 URL の平均更新間隔を算出した結果によると，1 回以上更新された URL では毎週更新されるものの割合が最も高くなっており，1 週間未満で更新される URL に関しては多くの情報が欠落してしまっていると推測される．一方，近年では Web サイトの更新情報を RSS 等のフィードにより積極的に通知することが一般的となってきた．フィードに対してはマナー上の問題をあまり引き起こさずに高頻度なアクセスを行えるものと考えられる．筆者らは大量のフィード URL へのアクセスを 1 年以上にわたってほぼ毎日続けており，そのログを Web ページ更新履歴の代替として用いることとした．

図 4 は利用したフィード更新ログのうち，2006 年 4 月から 2007 年 3 月までの 1 年間に複数回の更新が発生したフィード URL 163,122 件について，各 URL の隣接バージョン間の時間間隔 (each version) と URL ごとの平均更新間隔 (per-URL avg.) の分布を示したものである．ただし，以下のシミュレーションにおいては時間分解能を 1 日としたので，同日内に複数の更新が発生した際はそれらをまとめて 1 回として扱った．最小間隔である 1 日ごとに更新されるものの割合が最も高くなっており，更新間隔が大きくなるとともに power-law 的に頻度が減少している．

図 5 は更新間隔の最ゆう推定に基づくスケジューリング手法と増加減少制御による手法に対し，複数のパラメータを設定して 1 年分のシミュレーションを実行した際の最終的な平均更新網羅率及び平均アクセス効率を示したものである．水平軸の平均更新網羅率は取得した一意バージョン数と真のバージョン数の比を全 URL について平均したものであり，バージョンの取りこぼしの少なさを意味する．また，垂直軸の平均アクセス効率はアクセス回数に対する更新検出回数（一意バージョン数）の比を全 URL について平均したものであり，無駄なアクセスの少なさを意味する．

最ゆう推定に基づく手法のうち，$\bar{t_c}$ として $t_{c\min}$ を用いたものを MLE.min，$t_{c\,\mathrm{avg}}$ を用いたものを MLE.avg，$\sqrt{t_{c\min} \cdot t_{c\,\mathrm{avg}}}$ を用いたものを MLE.mix と表記している．MLE.mix は，偶然観測された値が影響し続ける最小値と，値の大きい側に振れやすい算術平均をブレンドし，適度な値を得ることを意図したものである．また，括弧内の 0.5–2，0.1–10 などは乗数の下限及び上限 μ_l–μ_h に対応し，続く 2nd は 2 回目のアクセスまでの間隔 τ_2 の値を表す．更に，x0.5 と記されたものは次回アクセス間隔と更新間隔推定値の比 α として 0.5 を用いたものであり，記載がないものは $\alpha = 1$ としている．

一方，AIMD については増分 d の値と減少係数 r がパラメータとなっている．増分が 1/2 日となっているものは，更新非検出アクセスが 2 回連続したときのみ増分 1 日を加算する変形手法である．

いずれの手法においても，網羅率は 0.9 程度と高く，効率が 0.5 周辺という結果になった．また，網羅率と効率はトレードオフになっており，飛び抜けて優位な

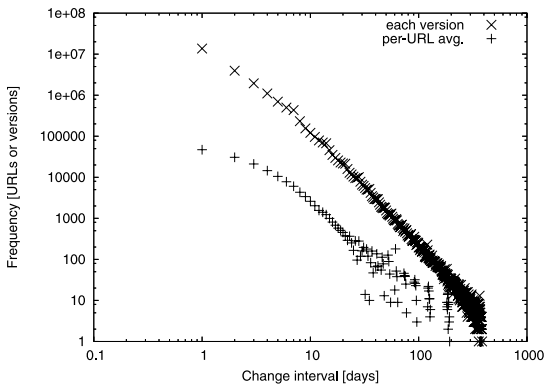

図 4　フィード更新ログにおける更新間隔の分布
Fig. 4　Distribution of change intervals in the feed update log.

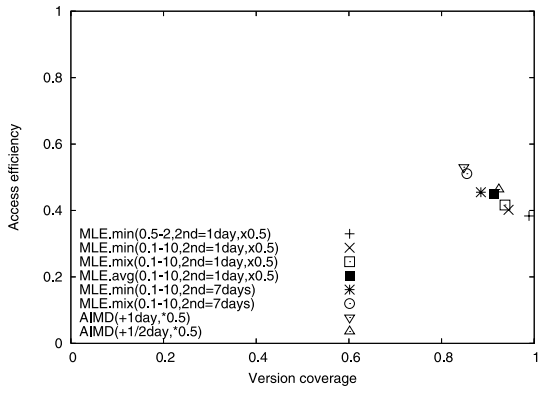

図 5　各スケジューリング手法の更新網羅率及びアクセス効率
Fig. 5　Version coverage and access efficiency of scheduling methods.

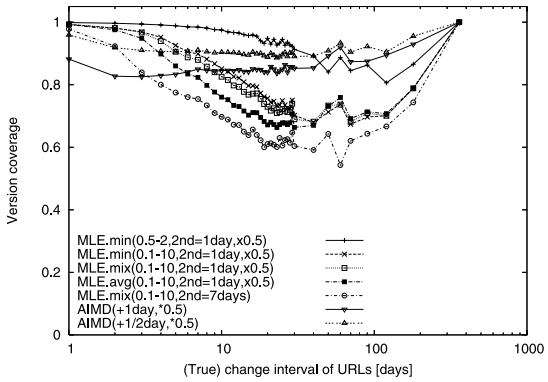

図 6　URL 平均更新間隔ごとの更新網羅率
Fig. 6　Version coverage per URL change interval.

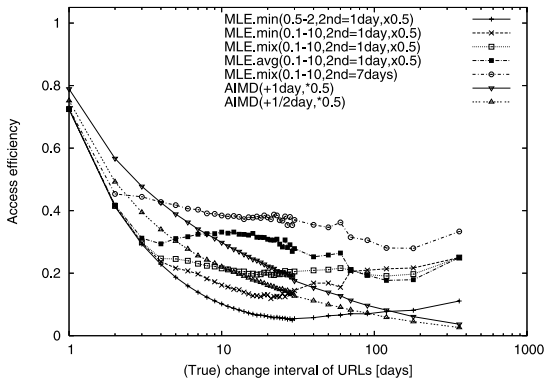

図 7　URL 平均更新間隔ごとのアクセス効率
Fig. 7　Access efficiency per URL change interval.

手法は見られない．ただし，図 5 ではすべての URL に関する結果を平均しているため，多数存在する小更新間隔 URL における傾向が強く反映されていることに注意が必要である．

図 6 及び図 7 は URL の真の平均更新間隔ごとに独立に求めた網羅率及び効率を示している．ただし，更新間隔が 30 日以上 100 日未満のデータについては 10 日ごと，更新間隔 100 日以上のデータについては 30 日ごとの平均値を用いた．更新間隔が大きな URL に対してはスケジューリング手法ごとの差異が明確になっている．乗数の範囲 0.1–10 の MLE では，URL の更新間隔が大きくなるにつれて網羅率及び効率が低下するが，効率は 0.2〜0.4 で安定する．MLE.mix は網羅率において同一パラメータの MLE.min を若干下回る程度であるが，効率においては明らかに上回っており，MLE.min や MLE.avg よりも好ましいことが読み取れる．

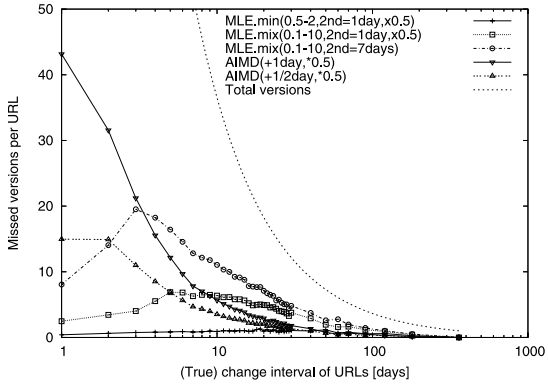

図 8　URL 平均更新間隔ごとの取得漏れバージョン数
Fig. 8　Missed versions per URL change interval.

一方，乗数の範囲 0.5–2 の MLE や AIMD では，網羅率は比較的安定しているものの，効率は 0.1 を下回るまで低下を続けている．これは，後者では次回アクセス間隔がとる値の範囲が小さく，大きな更新間隔への適応が不十分であることを示しており，高い網羅率はその副作用にすぎないといえる．フィードと比較して通常の Web ページでは更新間隔の長いものの割合が相対的に増し，全体の URL 数も増えると考えられるため，更新間隔への適応が不十分な手法は適さない．

MLE(0.1–10) では平均更新間隔が 60 日付近の URL に対して網羅率の低下が大きいが，平均更新間隔が大きくなると更新回数が減少し，1 回の更新を見逃す影響が増すことが一因と考えられる．図 8 は網羅率ではなく，取得できなかったバージョンの絶対数を示したものであり，Total versions で示した曲線が全バージョン数を表している．これによるとむしろ更新間隔が小さい URL に対する AIMD の漏れが重大であることが分かる．

図 9 は 1 か月ごとの更新網羅率の変化を追跡したものである．MLE の網羅率が時間とともに向上しているのに対し，AIMD は減少傾向にある．この点からも長期的な更新クローラに AIMD を用いるのは好ましくないといえる．

5. 2　実クローリングの状況と課題

ここでは更新クローラの動作状況の概要を述べる．更新クローラのスケジューリング手法には，クローリングの実施と並行して改良を加えたが，最終的に用いたのは前節の記法で MLE.mix(0.1–10,2nd=15days) に相当するものである．また，URL state DB の初期値には，過去数回分の一括クローリングの履歴を反映した．

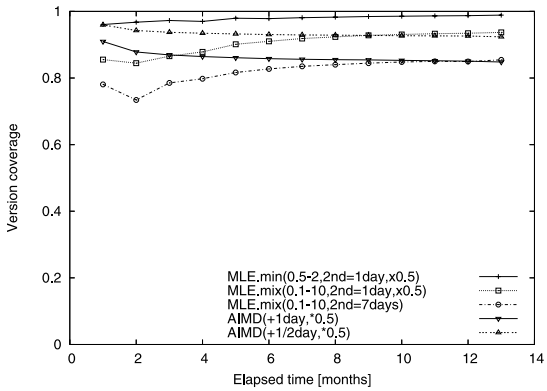

図 9　更新網羅率の時間変化
Fig. 9　Trace of version coverage.

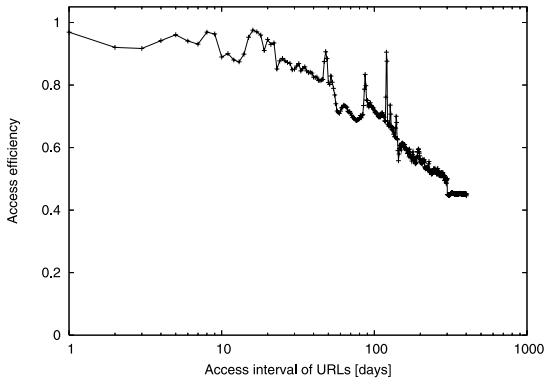

図 10　実クローリングにおける URL アクセス間隔ごとのアクセス効率
Fig. 10　Access efficiency per access interval of URLs in actual crawling.

図 10 は 2005 年 4 月から 2006 年 10 月にかけて複数回アクセスした Web ページのうち，約 3 億 URL について次回アクセス間隔ごとに求めた平均アクセス効率を示している．アクセス間隔約 1 年に相当するグラフの右端では効率は約 0.45 となっており，図 7 の右端における MLE.mix(0.1–10,2nd=7days) の効率 0.33 に接近しているが，アクセス間隔が小さい領域では両者の乖離が大きくなっている．URL ごとの真の更新タイミングが不明なため，更新網羅率を求めることはできないが，図 10 の左側では効率が 1 に非常に近くなっていることから，図 6 と比べ，低い値になっているものと考えられる．

実クローリングにおいては Web サーバに及ぼす負荷を抑える必要があり，多数の Web ページが属している Web サーバにおいてはスケジューリングどおりのタイミングで Web ページのアクセスを実行できないことが発生し得る（オーバフロー）[3]．また，クローラ自身の負荷がボトルネックとなる可能性もある．このような状況では更新網羅率の低下が避けられないが，むしろ意図的に目標とする更新網羅率を引き下げ，制御を回復する手法について検討を行っている．これにより，例えばある Web サーバに属する URL へのアクセス間隔を一律 2 倍とし，更新網羅率の目標を 1 から 0.5 に修正することで当該 Web サーバのオーバフロー状態を解消し，安定したクローリングを継続することが可能になると考えられる．

6. むすび

本論文では，Web 分析の基盤となる大規模 Web アーカイブの構築を目的とした更新クローラにおいて，高い時間分解能を得るための Web ページ再アクセススケジューリング手法を提案し，日次フィード更新ログを用いたシミュレーションによりその有効性を示した．

Web ページ再アクセススケジューリング手法は Poisson 過程における更新間隔の最ゆう推定に基づき，わずかな状態変数で記述できるよう簡略化を行った．これにより，大量の URL に対して個別に更新間隔の推定を行うことが可能となった．

Web アーカイブ構築においては収集情報の新鮮さよりも，過去まで含めたバージョン履歴の網羅性が重要となる．この観点での評価を可能にするため，日次アクセスにより得られたフィード更新ログを用いたシミュレーションを行った．その結果，1 日程度の小さな更新間隔の URL に対し，漏れの少ないスケジューリングが実現できていることを確認した．一方，アクセス間隔の修正幅に関するパラメータが小さいと，大きな更新間隔への適応に難があることが明らかになった．また，更新間隔の推定を行わず逐次的にアクセス間隔の調整を行う増加減少制御にも同様の問題があることが分かった．

今後は，実クローリングにおいて問題となる過負荷状態の解消手法について検討し，更新クローラへの統合を進めていきたい．

謝辞　本研究の一部は文部科学省リーディングプロジェクト e-Society 基盤ソフトウェアの総合開発「先進的なストレージ技術および Web 解析技術」による．

文　献

[1] 豊田正史，喜連川優，“日本におけるウェブコミュニティの発展過程，” 日本データベース学会 Letters, vol.2, no.1, pp.35–38, 2003.

[2] M. Toyoda and M. Kitsuregawa, “What’s really new on the web?: Identifying new pages from a series of unstable web snapshots,” Proc. WWW ’06, pp.233–241, 2006.

[3] 田村孝之，喜連川優，“大規模 Web アーカイブのための更新クローラの設計と実装，” DEWS2007 論文集，2007.

[4] International Internet Preservation Consortium, “netpreserve.org,” http://netpreserve.org/

[5] 国立国会図書館，“インターネット資源選択的蓄積実験事業 (WARP)，” http://warp.ndl.go.jp/

[6] 廣瀬信己，“国立国会図書館におけるウェブ・アーカイビングの実践と課題，” 情処学研報，vol.2003, no.51, pp.95–111, 2003.

[7] Internet Archive, “About IA,” http://www.archive.org/about/about.php

[8] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic, “Introduction to Heritrix, an archival quality web crawler,” Proc. IWAW, 2004.

[9] K. Sigurosson, “Incremental crawling with heritrix,” Proc. IWAW, 2005.

[10] K. Sigurosson, “Managing duplicates across sequential crawls,” Proc. IWAW, pp.99–114, 2006.

[11] B.E. Brewington and G. Cybenko, “How dynamic is the Web?,” Proc. WWW9, pp.257–276, 2000.

[12] J. Cho and H. Garcia-Molina, “The evolution of the web and implications for an incremental crawler,” Proc. VLDB, pp.200–209, 2000.

[13] J. Cho and H. Garcia-Molina, “Estimating frequency of change,” ACM TOIT, vol.3, no.3, pp.256–290, 2003.

[14] K.C. Sia and J. Cho, “Efficient monitoring algorithm for fast news alert,” Technical Report, UCLA, 2005.

[15] J. Edwards, K.S. McCurley, and J.A. Tomlin, “An adaptive model for optimizing performance of an incremental web crawler,” Proc. WWW, pp.106–113, 2001.

[16] J. Cho and H. Garcia-Molina, “Effective page refresh policies for web crawlers,” ACM TODS, vol.28, no.4, pp.390–426, 2003.

[17] J. Cho and A. Ntoulas, “Effective change detection using sampling,” Proc. VLDB, pp.514–525, 2002.

[18] 熊谷英樹，山名早人，“リンク構造を利用した Web ページの更新判別手法，” DEWS2004 論文集，2004.

[19] sitemaps.org, “Sitemaps XML format,” http://www.sitemaps.org/protocol.html

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext transfer protocol – HTTP/1.1,” RFC 2616, 1999.

[21] A.Z. Broder, S.C. Glassman, M.S. Manasse, and G. Zweig, “Syntactic clustering of the web,” Comput. Netw., vol.29, no.8-13, pp.1157–1166, 1997.

[22] M.S. Charikar, “Similarity estimation techniques from rounding algorithms,” Proc. STOC ’02, pp.380–388, 2002.

[23] M. Henzinger, “Finding near-duplicate web pages: A large-scale evaluation of algorithms,” Proc. SIGIR ’06, pp.284–291, 2006.

[24] L. Yi, B. Liu, and X. Li, “Eliminating noisy information in web pages for data mining,” Proc. SIGKDD ’03, pp.296–305, 2003.

[25] D. Chakrabarti, R. Kumar, and K. Punera, “Page-level template detection via isotonic smoothing,” Proc. WWW ’07, pp.61–70, 2007.

[26] D.M. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” Computer Networks and ISDN Systems, vol.17, no.1, pp.1–14, 1989.

[27] L. Qian, A. Quamruzzaman, and J. Attia, “Energy efficient sensing of non-cooperative events in wireless sensor networks,” Proc. CISS, pp.93–98, 2006.

[28] A. Ntoulas, J. Cho, and C. Olston, “What’s new on the web?: The evolution of the web from a search engine perspective,” Proc. WWW, pp.1–12, 2004.

（平成 19 年 5 月 31 日受付，9 月 28 日再受付）

田村　孝之　（正員）

1991 東大・工・電子卒．1996 同大大学院工学系研究科博士課程単位取得退学，NEDO 最先端分野技術研究員等を経て，1998 三菱電機（株）に入社．博士（工学）．現在，同社情報技術総合研究所専任並びに東大生産技術研究所協力研究員．並列データベース処理，Web マイニングに関する研究に従事．情報処理学会，日本データベース学会，ACM，IEEE Computer Society 各会員．

喜連川　優　（正員：フェロー）

昭 53 東大・工・電子卒．昭 58 同大大学院工学系研究科情報工学博士課程了．工博．同年同大生産技術研究所講師．現在，同教授．戦略情報融合国際研究センター長．データベース工学，並列処理，Web マイニングに関する研究に従事．情報処理学会フェロー，SNIA-Japan 顧問，ACM SIGMOD Japan Chapter Chair (’98〜’02)，平 9，10 本会データ工学研究専門委員会委員長．VLDB Trustee，IEEE ICDE，PAKDD，WAIM ステアリングコミティメンバ．