社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DBS－140 (II) (87)
2006／7／14

Webページ移動先発見のためのクローリング手法の提案

澤 菜津美† 飯田敏成† 森嶋厚行†
杉本重雄† 北川博之††

World Wide Webは我々の社会に不可欠なメディアとなりつつあるが，コンテンツが分散管理されていることから，しばしば内容の一貫性が維持されていないことがある．我々はWebコンテンツの一貫性維持の問題の一つとして，リンク切れの問題に取り組んできた．特に，リンク切れがWebページの移動によって引き起こされた時に，移動先の発見を行う問題に焦点を当ててきた．これまでの実験の結果，多くのページが，同一サイト内で移動していることがわかった．したがって，Webサイト中のWebページをクローリングすることが，移動先発見の手法として有効であるといえる．しかし，大規模なWebサイト全体を網羅的にクローリングすることはコスト的に問題がある．本稿では，クローリングにおけるページ訪問の順序を工夫することにより，より少ないページ数で移動先ページを発見するための手法を提案する．提案手法と深さ優先探索を実験で比較した結果，提案手法が有効であることがわかった．

Proposal of a Crawling Method for Finding Moved Web Pages

Natsumi Sawa,† Toshinari Iida,† Atsuyuki Morishima,†
Shigeo Sugimoto† and Hiroyuki Kitagawa ††

While the World Wide Web has become an indispensable medium in our society, the integrity of its contents is not always maintained because of its distributed architecture. We have been tackling the problem of fixing broken Web links, which is an example of the lost integrity of Web contents. In particular, we have been focusing on the problem of how to find moved Web pages when the movement causes broken Web links. Our previous experiments on the problem suggested that many moved Web pages can be found at the same Web site as the Web pages were originally located. Therefore, crawling through the Web site is an effective way to find moved Web pages. An exhaustive crawling, however, would take a huge cost when the size of the Web site is large. This paper proposes a crawling algorithm that visits Web pages in an efficient order. We compared our algorithm with the depth-first order crawling and found that our algorithm is effective.

1. はじめに

近年，World Wide Web (以下Web) は社会における重要なメディアの一つとして大きな役割を果たしている．Webの特徴の一つとして分散管理が行われていることが挙げられる．この特徴は，Webを便利なツールとする一方で，Webコンテンツの一貫性の維持を困難としている要因でもある．コンテンツの一貫性が損なわれる一例としてWebのリンク切れがあり，我々はこれに着目している．

我々はWebページの移動先を発見し，ページ移動によるリンク切れの自動修正を試みるシステム (WISHシステムと呼ぶ) を開発して，実験を行ってきた[1)2)3)4)]．図1はWISHシステムのアーキテクチャである．処理の流れは次の通りである．あらかじめ，ユーザがシステムにリンク u を登録しておくと，システムは登録された u を監視する．同時に，ページコンテンツのキャッシュ c を保存する．システムがリンク切れを発見すると，ページ探索モジュール (Page Chaser) が，移動先の探索を始める．探索の結果は「ページの移動先らしさ」でランキングされたURLのリスト $\bar{U}$ であり，利用者に返される．

これまでの実験から，Webページの移動先の多くが同一サイト内であることが判明した[3)]．WISHシステムでは，同一サイト内の移動ページを発見するためのクローリングを行っている．しかし，大規模なWebサイト全体を網羅的にクローリングすることはコスト

† 筑波大学大学院 図書館情報メディア研究科
Grad. Sch. of Info. and Media Studies, Univ. of Tsukuba.
†† 筑波大学大学院 システム情報工学研究科
Grad. Sch. of Sys. and Info. Eng., Univ. of Tsukuba.

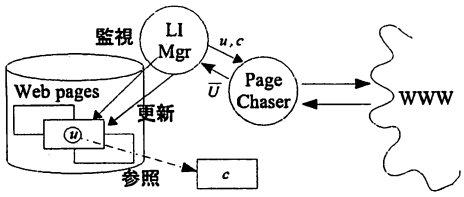

図 1 WISH システム

的に問題がある．本稿では，ページの移動先の場所に偏りがあることに着目し，クローリングにおけるページ訪問の順序を工夫して，より少ないページ数で移動先ページを発見するための手法を提案する．

本稿は次のように構成されている．まず，2 章で関連研究を説明する．3 章で提案手法を説明する．4 章では，実験により深さ優先探索と提案手法の比較を行う．5 章はまとめと今後の課題である．

2. 関連研究

クローリングの研究は数多くあるが，基本的には「ある基準を満たすページを網羅的に収集する」ことを目的としている．例えば，次のようなものがある．(1) インデックスサーバのためのクローリング: Google や Yahoo!などの検索エンジンではクローリングによって Web ページを収集している．目的は網羅的収集であるが，効率の良い収集のための工夫が行われている．例えば Google のクローリングでは，PageRank の高いページから優先して収集する手法[5] を使用している．他にも，更新された Web ページを優先的に収集する手法[6] などがある．(2) 特定の用途に特化したクローリング: 地域情報収集のためのクローリング手法[7] では，ある地域に関する情報だけを収集することに特化したクローリング手法を提案している．

本研究がこれらと根本的に異なる点は，「ある特定のページを発見するためのクローリング」であり，多くのページの網羅性は一切必要ないことである．したがって，必要なページを発見さえできれば，必ずしも網羅探索を行う必要はない．

3. リンク経路を利用したクローリング手法

本章では，本手法で利用する「リンク経路」について説明した後，どのようにリンク経路を求め，リンク経路を利用したクローリングを実現するのかについて述べる．

3.1 リンク経路

定義. 各 Web ページの URL を u_i，Web リンクを有向辺 (u_i, u_j) とし，Web を有向グラフでモデル化する．この時，u_0 から u_m へのリンク経路 (link path) とは $u_0 \rightarrow u_1 \rightarrow \cdots \rightarrow u_m$ の形をした Web リンクの有限列のうち，URL 群 $u_0, u_1, \ldots, u_m$ が全て異なるものである (ただし，$u_0 = u_m$ であってもよい)．

本手法では，図 2 に表すような Web サイトのルートページ u_0 から監視対象ページ u_{end} までの 2 点間のリンク経路の情報を利用する．本手法のポイントは，ページの移動先はリンク経路の周辺に存在する可能性が高いというヒューリスティクスを利用し，その周辺を優先してクローリングを行うことである．

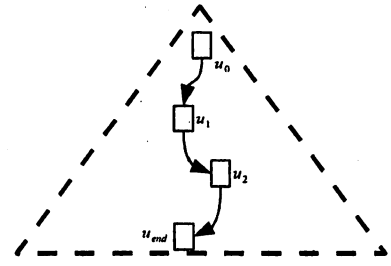

図 2 本手法で利用するリンク経路

本手法は，下記の 2 つのフェーズから構成される．

[フェーズ 1] リンク経路発見. これはリンク u_{end} が監視対象として WISH システムに登録された時点で行われる．

[フェーズ 2] リンク経路を利用したクローリング. これは u_{end} がリンク切れしたときに行われる．

3.2 フェーズ 1: リンク経路発見アルゴリズム

リンク経路発見は，監視対象リンクが指す URL u_{end} を手がかりとして，ルートページ u_0 から u_{end} までのリンク経路を求める．探索の流れはリンクの向きとは逆になる．初めに u_{end} にリンクする u_{end-1} を求め，次に u_{end-1} にリンクする u_{end-2} を求め…，と続き最後に u_0 が求められると，リンク経路完成となる．

図 3 がリンク経路探索のアルゴリズムである．関数 PathFind は入力として，監視対象リンクが指す URL である u_{end} を受け取る．URL のリスト P を $[u_{end}]$ に初期化する (8 行目)．P はリンク経路探索の中間結果を保持するためのデータ構造である．URL u_{end} を構成する文字列からディレクトリ部分を削除し，u_0 を求める (9 行目)．u_{end} をディレクトリ単位で区切った複数の URL(ただし，ルートページの URL は除く) を $u_{(0<i<end)}$ の候補の集合 *candidates* に加える (11 行目)．例えば，http://a/b/c/dog.html という URL の場合，http://a/b/,http://a/b/c/が *candidates* に加えられる．PathFind から呼び出される関数 Path-

Grow によって，サイトのルートに向かってリンク経路を構築していく．

関数 PathGrow は次のように動作する．すなわち，PathFind から最初に呼ばれたときは，リンク経路の中間結果は $P = [e]$ である．そして，PathGrow の再帰呼び出しによって，あるパス $P = [u_0, u_1, \ldots, u_{n-1}, u_{end}]$ を構築する．$P = [u_0, u_1, \ldots, u_{n-1}, u_{end}]$ が一つ構築されるたびに，P の集合である *path* に保存され，次の P の探索へと移る．このように，複数のリンク経路が (存在すれば) 発見されることになる．Web ページのアクセス数を制限したいときには，定数 *max* に適切な値を入れておけば，*max* 回のページアクセスを行った時点でアルゴリムは終了する．

関数 PathGrow の詳細は次の通りである．

18-24 行目 P の先頭から URLu_1 を取り出し，まだ訪問していないページならば，関数 *getAnchor* により u_1 のコンテンツから URLu_i およびアンカー文字列を抽出する．抽出した u_i は集合 *candidates* に入れる．この *candidates* には，これまでの探索で収集した u_i が全て格納されている．

25-37 行目 各 $u_i \in candidates$ に対し，u_i から u_1 へリンクしているかどうかを調べる．true ならば，さらに，$u_i = u_0$ かどうか調べる．$u_i = u_0$ ならばリンク経路完成となり，P は *path* へ格納される．$u_i \neq u_0$ ならば，u_i' として集合 *upper_set* に格納される．

39-43 行目 各 $u_i' \in upper\_set$ に対し，“論理的上位ページらしさ” を判定し，スコア $score_i$ を求め，*upper_set* に格納する．ここで論理的上位ページについて説明する．Web リンク $u_1 \rightarrow u_2$ において，u_2 が u_1 に所属するとき，または，u_2 が u_1 の一部であるとき，u_1 を u_2 の論理的上位ページとよぶことにする．しかし，実際の Web ページ間で厳密に論理的上位ページの判定処理を行うのは容易ではないため，本手法では URL の構造に着目した以下の 3 つの単純な評価方法を用いている．

(1) u_1 のディレクトリ数と u_i' のディレクトリ数の差分を求める．URL はスラッシュで区切ってディレクトリの構造を表している．例えば，http://a/b/c/dog.html という URL の場合はディレクトリ数は 3 となる．u_1 よりディレクトリが少なければポイントは高くなり，多ければポイントは低くなる．ディレクトリ数が同じときは u_1 と同一ディレクトリであればその URL が優先される．

(2) ファイル名に index が含まれているとポイントが与えられる．

(3) u_i' と u_1 が相互リンクだとポイントが与えられる．

44 行目 関数 *sort* は *upper_set* に含まれる u_i' を $score_i$ 順でランキングし，リスト *ranking* に格納する．

45-50 行目 *ranking* の先頭から一つ u_i' を取り出し，これを P の先頭に追加する．そして，再帰的に PathGrow を呼び出す．

探索が終了すると，*path* に格納された URL と各ページに対応するアンカー文字列が出力される．また，P が完成しなかったときも，途中までの P が出力される．

3.3 フェーズ 2: リンク経路情報を利用したクローリングアルゴリズム

リンク切れが発生すると，リンク経路探索によって発見されたリンク経路 P を利用して，同一サイト内で移動先ページ v_{end} を探索する．探索の流れは，図 4 に示すように，まず P 周辺の探索を行い (図 4(i))，次に u_0 から未探索範囲に対して深さ優先の網羅探索を行う (図 4(ii))．

図 5 がリンク経路を利用したクローリングアルゴリズムである．関数 Crawl は，入力として，リンク経路の集合 *path* を受け取る．そして，監視対象ページの移動先候補のリスト E を出力とする．説明のため，u_0 と u_{end} を除いたリンク経路の URL のリストを P，P の任意のページを u_i とする．探索最大ページ数 *page* を設定し，移動先候補の集合 E に格納される移動先 e_i の総数が *page* に達するまで探索する．関数 Crawl の詳細は次の通りである．

9-14 行目 リンク経路から周辺探索を開始する．u_{end} から u_0 に向かって P をたどり，関数 *depth_first* は各 u_i から 1 リンク先のリンクを収集する．探索ページ総数が *page* に達した時点で探索を終了する．

16-18 行目 リンク経路からの探索が終了した時点で，u_i の総数が *page* に達していない場合は，u_0 から未探索範囲に対して，関数 *depth_first* は深さ優先探索を行い，リンクを収集する．

4. 実　　験

3 章で説明したアルゴリズムを実装し，本手法による探索と深さ優先探索とでそれぞれ移動先ページの探

```
Input: リンク経路のリスト P
Output: リンク経路の集合 path
 1    Global Set candidates={}; // u_i の候補の集合
 2    // リンク抽出が終わった URL の集合
 3    Global Set searched_set={};
 4    Global Set path={}; // 完成したリンク経路の集合
 5    Global int max=PathGrow を呼び出す最大回数;
 6    Global URL u_0; // ルートページの URL

 7    void PathFind(u_end){
 8      List P = [u_end]; // P に u_end をセット
 9      u_0 = get_root(u_end); // u_end から u_0 を抽出する
10      // u_end をディレクトリでカットして追加
11      candidates.addAll(cut_dir(u_end));
12      PathGrow(P);
13    }

14    void PathGrow(P) { // P を求める
15      max を 1 減らす;
16      if(max > 0) {
17
18        URL u_1 = P.getFirst(); //P から先頭を取り出す
19        // u_1 のコンテンツからリンクを抽出
20        if (!searched_set.contains(u_1)) {
21          // 抽出した URL を格納
22          candidates.addAll(getAnchor(u_1));
23          searched_set.add(u_1);
24        }
25        // candidates から u_1 へリンクする u'_i を抽出する
26        Set upper_set={}; // 抽出した u'_i の集合
27        for each u_i in candidates {
28          // u_i が u_1 にリンクしているかどうかを調べる
29          if (u_i → u_1) {
30            if (u_i = s) { // u_i が u_0 ならばリンク経路完成
31              path に P を格納; // 完成したリンク経路を記録
32            } else {
33              // u'_i 候補として upper_set に追加
34              upper_set.add(u_i,null);
35            }
36          }
37        }
38
39        for each u'_i in upper_set {
40          // 論理的上位ページらしさを判定
41          score_i = Check(u'_i,u_1);
42          upper_set.add(u'_i,score_i); // スコアを格納
43        }
44        List ranking = sort(upper_set);
45        // 再帰呼び出し
46        for each u'_i in ranking {
47          P.addFirst(u'_i);
48          PathGrow(P);
49          P.removeFirst();
50        }
51      }
52      return;
53    }
```

図 3　フェーズ 1: リンク経路発見アルゴリズム

索を行い比較検証した．また，アンカー文字列の利用の効果を検証するための実験を行った．

4.1 実　　験

実験では同一サイト内でのページ移動によるリンク切れページに対して，移動先の探索を行う．実験する探索方法は下記の二つである．

手法 (a) リンク経路を用いた探索: 3 章で提案した手法を用いてクローリングを行う．

手法 (b) 深さ優先探索: Web サイトのルートページ

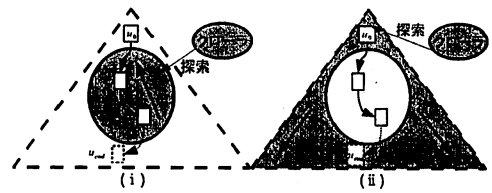

図 4　リンク経路を利用したクローリングアルゴリズム

```
Input:リンク経路の集合 path
 1    void Crawl(path){
 2      Set path={P_1,P_2,…,P_n}; // リンク経路の集合
 3      Set E={}; // 移動先候補の集合
 4      URL u_0=ルートページの URL;
 5      URL u_end=移動前のページの URL;
 6      int page=探索最大ページ数;
 7      int Lv=1; // パス周辺探索の深さ
 8      // リンク経路探索
 9      for each P_i in path {
10        for each u_i in P_i {
11          if(v_i の総数 >= page) break;
12          E.add(depth_first(Lv, u_i)); // パス周辺探索
13        }
14      }
15      //ルートページから深さ優先の網羅探索
16      if(v_i の総数 < page){
17        E.add(depth_first(null, u_0));
18      }
19    }
```

図 5　フェーズ 2: クローリングアルゴリズム

から探索を開始し，深さ優先のクローリングを行う．

実験方法についての詳細は次の通りである．本実験の探索対象は同一サイト内での移動によるリンク切れの移動先ページである．対象ページは，既にページ移動によるリンク切れが発生しており，WISH システムで同一サイト内で移動先が発見されているページを利用した．総数は 30 ページである．本来の提案手法の利用では，リンク経路探索はページが移動する前に行われる．しかし，本実験では既にリンク切れ発生後であるため，リンク切れ発生前の過去のサイト構造でリンク経路の発見を行わなければならない．したがって，本実験ではリンク経路発見のために，インターネットアーカイブ[8] を使用した．その後，発見されたリンク経路を用いて，現在の Web サイト構造での移動先探索を行った．

リンク経路発見のための最大探索リンク数パラメータである max は 10 に設定した．また，移動先ページ探索時に訪問するページ数の上限を 1000 として探索を行った．後述するように，本実験において，手法 (a) では 1000 ページ以内に全て発見できたが，手法 (b) では，1000 ページを訪問しても発見できなかった移動先が存在する．

移動したページ 30 ページそれぞれの移動前の URL

を u とする．その移動先の URL を v とする．このとき，ある手法が移動先 v を訪問した順番を $n(v)$ とする．また，u と v が含まれる Web サイトの総 HTML ページ数を N_u とする．このとき，下記で定義される $V(u)$ を計算する．

$$V(u) = \frac{n(v) \times 100}{N_u}$$

すなわち，$V(u)$ は，u の移動先を発見するために，サイトに含まれる総ページ数のうち何パーセントのページを訪問する必要があったか，という割合である．

ページ u が含まれる Web サイトのサイズが大きかったため，本実験では，N_u の値は下記の手法を用いて近似的に求めた．すなわち，Google のドメイン指定検索とファイル指定検索 (html および htm ファイル) を組み合わせた検索結果数を N_u とした．

4.2 実験の結果

すべての移動ページ u_i に対して $V(u_i)$ を求め，その結果を累積度数分布図にまとめると図 6 のようになった．図 6 において，x 軸は $V(u)$ が示す探索ページ数の割合 (%) を，y 軸は 30 ページのうち何割のページが発見できたかを表す発見率 (%) である．図 6 の (a) は，手法 (a) において，リンク経路を利用した探索フェーズにおいて訪問したページ数である．(a') は，リンク経路発見のための訪問ページ数を $V(u)$ の計算時に分子に加えた値である．今回は収集するページ数の上限を 1000 と限定したため，手法 (b) による探索では移動先ページが収集されなかったものが 8 個（実験数全体のおよそ 25%）存在した．

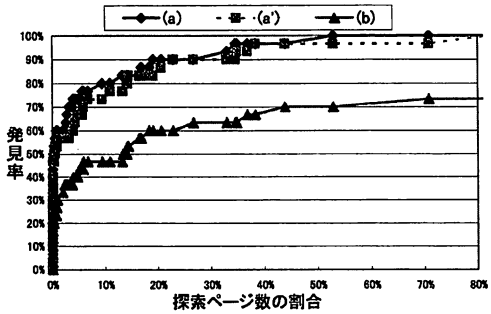

図 6 実験結果

図 6 において，発見率 70%での手法 (a) と (b) の探索ページ数を比較すると，(a) の場合は探索ページ数が約 3%であるのに対し，(b) では約 43%となった．この場合，(a) は (b) の 1/10 以下の探索量で，同じ発見率を実現することができた．グラフ全体を見ても，手法 (a) の発見率が手法 (b) の発見率を大きく上回ったことがわかる．また，図において (a) と (a') のグラフがほぼ並んでいることから，リンク経路発見の探索量のコストが低いことがわかる．

4.3 考　　察

実験の結果から，リンク経路は移動先探索の有効な手がかりとなるとわかった．本稿では詳細な分析結果は省略するが，特に，リンク経路上のページが監視対象のページが移動した後も存在している場合は，リンク経路上のページは移動先探索の非常に有力な手がかりとなり，リンク経路上のページから 1 リンク先までの探索で移動先ページにたどり着くことができた．

実験の (a) と (a') を比較すると，低いコストで，移動先探索に効果的なリンク経路情報を発見できたことがわかる．一般には，Web サイトのルートから対象ページまでのリンク経路は複数存在しており，必ずしもリンク経路の利用が効果的とは限らないと考えられるため，この結果は興味深い．今回の実験で，効果的なリンク経路を低コストで発見できた理由は，リンク経路の発見手法にあると考えられる．すなわち，本手法では移動前のページのコンテンツと URL を手がかりにしてボトムアップにリンク経路を探索していく際に，ただリンクが張られているかどうかだけではなく，“論理的上位ページらしさ” を判断しながら探索を行う．その結果，より移動前ページと関連すると思われるリンク経路から優先的に見つけることが出来た．そのため，有効なリンク経路を優先して見つけることができたと考えられる．

5. まとめと今後の課題

本稿では，同一サイト内での Web ページの移動先を効率的に発見する手法として，リンク経路を用いた探索を提案した．実験を行った結果，本手法は深さ優先探索よりも少ない探索量で移動先ページを発見することを確認できた．したがって，リンク経路を利用する方法は移動先の発見に有効であり，探索量を減らしても探索もれの可能性が下がることがわかった．

今後の課題は以下の通りである．まず，今回は必要なパラメータの値を固定して実験を行ったが，これらの値は発見率および探索量に影響するため，パラメータに関する分析が必要である．次に，探索の順番に関してさらなる工夫を行い，より効率のよいアルゴリズムを開発することがあげられる．最後に，同一サイト以外のサイトに移動したページを効率よく発見するためのクローリング手法の開発がある．

[謝　辞]

ゼミなどでご議論いただきました筑波大学大学院図書館情報メディア研究科の阪口哲男助教授，永森光晴講師に感謝致します．本研究の一部は科学研究費補助金特定領域研究（#18049005）による．

参 考 文 献

1) 中溝昌佳, 森嶋厚行, 有山智洋, 杉本重雄, 北川博之, WWW コンテンツ一貫性維持のためのリンク更新機構の提案. 日本データベース学会 Letters, Vol.2, No.2, pp.65-68, 2003 年 10 月.

2) 中溝昌佳, 森嶋厚行, 杉本重雄, 北川博之, WWW における信頼度の高いリンクの発見. 情報処理学会研究報告 Vol.2004, No.72(2004-DBS-134(II)), pp.397-402. 電子情報通信学会技術研究報告 Vol.104, No.177(DE2004-63), pp.87-92, 2004 年 7 月.

3) 中溝昌佳, 森嶋厚行, 杉本重雄, 北川博之, WWW リンク一貫性維持支援システムにおけるリンク切れ自動修復. 日本データベース学会 Letters, Vol.3, No.3, pp.5-8, 2004 年 12 月.

4) 中溝昌佳, 飯田敏成, 森嶋厚行, 杉本重雄, 北川博之, Web リンク切れの自動修正における信頼度の高いリンク情報の利用. 電子情報通信学会第 16 回データ工学ワークショップ (DEWS2005), 7 pages, 2005 年 3 月.

5) Junghoo Cho, Hector Garcia-Molina, Lawrence Page, Efficient Crawling Through URL Ordering. Computer Networks,Vol. 30,No. 1-7,pp.161-172,1998.

6) 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. 電子情報通信学会第 15 回データ工学ワークショップ (DEWS2006), 2004 年 3 月.

7) 張建偉, 石川佳治, 黒川沙弓, 北川博之, 地域ウェブ情報源の収集のためのクローリング手法の提案, 電子情報通信学会第 16 回データ工学ワークショップ (DEWS2005), 2005 年 2 月.

8) Internet Archive. http://www.archive.org/.