リンク構造解析アルゴリズム高速化のための縮小 Web の構築

Web-Link Structure Reduction for Accelerating Link Analysis Algorithms

片瀬 弘晶♥ 松永 拓♥ 上田 高徳♥
田代 崇♥ 平手 勇宇♦ 山名 早人♣

Hiroaki KATASE Taku MATSUNAGA Takanori UEDA
Takashi Tashiro Yu HIRATE Hayato Yamana

本稿で提案する縮小 Web は，Web グラフ内のノードをクラスタリングし，ノード数・エッジ数を削減することにより得られる．この縮小 Web は，Web グラフに直接既存のリンク構造解析アルゴリズムを適用して得られる結果を近似的に得る代わりに，計算量およびグラフのデータ容量を削減することができる．我々は，Web のリンクデータを繰り返し利用する際のデータ容量的・計算コスト的な負荷を軽減するアプローチの一つとして，縮小 Web を考案した．

We propose "Little Web" which can be constructed from Web Graph by clustering nodes in Web Graph. In case of using link analysis algorithm for Little Web, its result will be approximate against a case of original Web Graph, but the computational cost and data size of link analysis algorithm with Little Web can be reduced. We devised Little Web as one of the approaches of decreasing computational cost and data size when repeatedly using link data.

1. はじめに

Web 上の広範囲にわたる Web ページを対象とし，ページに含まれるハイパーリンクから構築される Web グラフを解析するには，相応の計算資源と計算量が必要となり，また Web グラフ自体を保存するために要する記憶容量も大きなものとなる．この問題に対し，我々は Web のリンク構造を利用したランキングアルゴリズムやコミュニティ抽出アルゴリズムなどのリンク構造解析アルゴリズムの多くがグラフ構造のノード数・エッジ数に依存して計算量が増加するという事実に注目し，Web グラフのリンク構造を圧縮することでノード数・エッジ数を減らし，計算量の軽減とグラフ構造の保存に必要な記憶領域の削減が可能になるのではないかと考えた．そして，本稿において，実際に既存のリンク構造解析アルゴリズムの高速化とグラフ構造データの削減を目的とする Web グラフのノード数とエッジ数を削減したグラフ構造(以降，縮小 Web)の構築手法を提案する．縮小 Web は，Web グラフに対して一度しかリンク構造解析アルゴリズムを適応しない場合には利点がないが，Web グラフに対して繰り返しリンク構造解析アルゴリズムを適用する場合や，いくつかの異なるリンク構造解析アルゴリズムを用いる場合に，計算量的優位性を得ることができるものとなる．

本稿では以降，第 2 節で関連研究を示し，第 3 節で，提案手法である縮小 Web の構築手法を記す．そして第 4 節で縮小 Web の評価について述べ，第 5 節でまとめを行う．

♥ 学生会員 早稲田大学大学院基幹理工学研究科情報理工学専攻修士課程
{katase,taku,ueda,ttashiro}@yama.info.waseda.ac.jp
♦ 正会員 早稲田大学メディアネットワークセンター
hirate@yama.info.waseda.ac.jp
♣ 正会員 早稲田大学理工学術院
yamana@info.waseda.ac.jp

2. 関連研究

縮小 Web は Web グラフの圧縮を利用してリンク構造解析アルゴリズムの計算コストを削減する．本節では，リンク構造解析アルゴリズムと Web グラフの圧縮に関連する研究を取り上げる．

2.1. リンク構造解析アルゴリズムの高速化

リンク構造解析アルゴリズムの一種である HITS アルゴリズム[1]，PageRank アルゴリズム[2]に対してアルゴリズムの高速化に関連する研究が行われている[3][4][5][6][7][8]．Kamvar らの研究[5][6][7]では，Web リンク構造を外挿法(Extrapolation)により推定することで計算コストを下げる手法，PageRank の初期値に同一ドメイン内のリンク構造のみから得られる PageRank を利用することで，収束までの反復回数を減少させる手法，PageRank の計算において早期に収束する箇所を除去することによる高速化の手法などが提案されている．Gollapudi らの研究[8]では，Bloom Filter を利用して与えられたクエリを含むページ群のリンク先 Web ページ，および被リンク元 Web ページ集合を，近似的にではあるが高速に取得することで，HITS，SALSA アルゴリズム[9]によるランキング計算時間を短縮させる手法が提案されている．

上記に挙げた高速化アルゴリズムは，特定のリンク構造解析アルゴリズムに特化したものである．これに対し，本研究の縮小 Web は，異なる種類のリンク構造解析アルゴリズムの高速化の実現を目標としている．また，縮小 Web は Web グラフからノード数・エッジ数を削減し構成されたものであるため，リンク構造解析アルゴリズムが実行に必要とする物理メモリ容量を削減する効果がある．これにより，状況によっては，本来計算機の物理メモリ上のみで実行困難であるアルゴリズムを，物理メモリ容量内で実行させることを可能にする．

2.2. Web グラフの圧縮

Buehrer らの研究[10]では架空のノード(以下，VN)を Web グラフに追加することで Web グラフのエッジ数を減少させ，グラフを圧縮する手法が提案されている．VN は図 1(a)のような 2 部グラフの中間に図 1(b)のような形で挿入される．Buehrer らは頻出パターンマイニングによって各 Web ページがリンクしている Web ページの共通項を抽出し，得られた頻出パターンをもとに，図 1(b)のように VN を挿入する．これによって Web グラフ全体のエッジ数を減少させている．また，[10]ではさらに，符号化を用いてノードの隣接リストが必要とするビット数の削減も行われている．

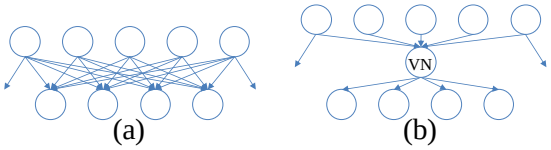


図 1 VN による Web グラフの圧縮

この手法により Buehrer らは 8 個中 7 個ほどの割合でエッジを削減できたとしている．エッジの圧縮率に関しては後に 4.3.3 で示す本手法の圧縮率よりも高い値を示している．ただ

し，この手法で圧縮された Web グラフに対して単純にリンク構造解析アルゴリズムを適用する場合，VN に張り替える際にまとめられたエッジと，まとめられなかったエッジを同等に扱ってしまうため，得られる解析結果に大きな影響が出る恐れがある．また，VN を追加することによる Web グラフの圧縮であるため，ノードの数は圧縮できていない点が，本研究と異なる．

3. 提案手法

3.1. 縮小 Web について

縮小 Web は，もととなる Web グラフ内のノードを，当該ノードのリンク先のノード集合(子ノード集合)と当該ノードへリンクを持つノード集合(親ノード集合)との類似性をもとにクラスタリングすることで得る．本稿では，この類似性を評価するための指標として HITS[1]の Authority と Hub を利用する．ノードの類似性の評価に Authority と Hub を利用する理由は，HITS の特性上それぞれが親ノード集合・子ノード集合に強く依存した値となっているからである．また，ノードを子・親ノード集合に基づいてクラスタリングを行う理由は，検索システムで利用されている PageRank，HITS，SALSA[9]や，コミュニティ抽出アルゴリズムが，しばしばノードの持つ情報として子・親ノード集合を重視することを考慮し，この情報の欠落を抑えるためである．

Web グラフのクラスタリングで得られたクラスタは，縮小 Web のグラフ構造におけるノードとなり，縮小 Web のエッジはクラスタ間に張られたものとなる．ただし，縮小 Web のエッジはクラスタリング結果をもとにして得られる重みを持つ．

3.2. 縮小 Web 構築手法

ノードのクラスタリングに用いるノードの類似性判定には，3.1 項で述べたように Authority と Hub を利用し，図 2 に示す関数 comp()を用いる．なお，comp()に記述されている αは 0 での除算を防ぐパラメータである．また，u, v は Web グラフのノードを表し，$auth[u]$ (≤1), $hub[u]$ (≤1)はそれぞれノード u の Authority, Hub である．

縮小 Web の構築は，図 3 に示す makeLW()によって行う．makeLW()は引数として Web グラフ $G(V, E)$を受け取り(V, E はそれぞれ G のノード集合，エッジ集合)，縮小 Web グラフ G_{LW}を返す関数である．makeLW()において，V_{LW}, E_{LW}はそれぞれ縮小 Web のノード集合，エッジ集合を表す．ここで，縮小 Web のノードはすべて，図 4 の関数 cluster()によってもととなる Web グラフのノードによって構成されるクラスタとして生成されるので，混乱を避けるために以降では縮小 Web のノードをクラスタと称すこととする．さらに，図 3 ～図 5 において，$C[u]$はノード u の含まれるクラスタ，$e(u, v)$はノード u からノード v へのエッジ，$e(C[u], C[v])$はクラスタ $C[u]$から $C[v]$へのエッジを表し，$EdgeW[e(C[u], C[v])]$はエッジ $e(C[u], C[v])$に割り振られている重みを表す．

makeLW()ではまず，図 4 の関数 cluster()を用いて V に含まれるノードをクラスタリングする．cluster()ではクラスタリングを行う前に，V に含まれる各々のノードをノード単体で構成されるクラスタとしてクラスタ集合 C に挿入する(図 4:1~3 行)．次に，E に含まれる各々のエッジの両端のノードについて併合が可能かどうかを判断するために，comp()を用いて類似度を計算し，類似度が高い順にエッジを並び替える(図 4:4~5 行)．その後、類似度が T_E以上となるクラスタを併合する (図 4:6~13 行)．2 つのクラスタが持つノード数がともに 1 である場合には，この 2 つのクラスタを併合する(図 4:9~10 行)．そうでない場合，図 5 の関数 judgeMerging()によって判定する(図 4:11~13 行)．併合の判定方法は judgeMerging()に記した擬似コードの通りである． judgeMerging()は併合を行った場合のクラスタが「クラスタ内の任意の 2 つのノードの類似度が閾値 T_C 以上」である場合にのみ真を返す関数である．

cluster()により縮小 Web のノード集合となる V_{LW}を得たら，続いて縮小 Web のエッジ集合 E_{LW}を生成する(図 3:4～9 行)．E_{LW}は，E に含まれるエッジを両端のノードが属するそれぞれのクラスタからクラスタへのエッジに変換して得られるエッジの集合となる(図 3:6 行)．このとき，エッジの両端が

```
comp(u,v)
1 :   if  auth[u] < auth[v]
2 :       x = (auth[u] + α) / (auth[v] + α)
3 :   else
4 :       x = (auth[v] + α) / (auth[u] + α)
5 :   if  hub[u] < hub[v]
6 :       y = (hub[u] + α) / (hub[v] + α)
7 :   else
8 :       y = (hub[v] + α) / (hub[u] + α)
9 :   return  x * y
```

図 2 類似度評価関数

```
makeLW(G(V, E))
 1 :   set V_LW ← cluster(V, E)
 2 :   set E_LW = ∅
 3 :   set EdgeW= ∅
 4 :   for each e(u, v) ∈ E
 5 :       if e(C[u], C[v]) ∉ E_LW
 6 :           E_LW ← E_LW ∪ {e(C[u], C[v])}
 7 :           EdgeW[e(C[u], C[v])] = 1
 8 :       else
 9 :           EdgeW[e(C[u], C[v])] += 1
10 :   return G_LW (V_LW, E_LW, EdgeW)
```

図 3 縮小 Web を構築する関数

```
cluster(V, E)
 1 :   set C = ∅
 2 :   for each u ∈ V
 3 :       C ← C ∪ {{u}}
 4 :   array E' ← { e(u, v) | e(u, v) ∈ E }
       // E'の要素e(u, v)をcomp(u, v)の値で降順に並び替える
 5 :   sort  E'  by comp()
       // E'の要素を5行で整列された順に処理
 6 :   for each e(u, v) ∈ E'
 7 :       if comp(u, v) < T_E
 8 :           break
 9 :       else if |C[u]| = 1 and |C[v]| = 1
10 :           C ← C ∪ {{u, v}}
11 :       else if judgeMerging(C[u], C[v])
12 :           C[u] ← C[u] ∪ C[v]
13 :           remove C[v] from C
14 :   return C
```

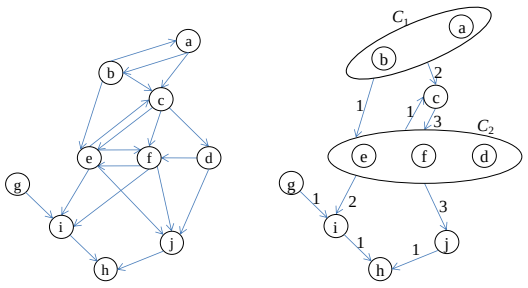
図 4 クラスタ集合 V_{LW} を構築する関数

```
judgeMerging(C_1, C_2)
1 :   for each u ∈ C_1
2 :       for each v ∈ C_2
3 :           if comp(u, v) < T_C
4 :               return false
5 :   return true
```

図 5 クラスタ併合判定関数

同一のクラスタとなるエッジが出現するが，これらエッジは除去せずに残しておく．このように処置しておく理由は，PageRank や HITS におけるランダムサーファーモデルを考慮し，ネットユーザが同一のクラスタにとどまる確率を定義できるようにするためである．また，E_{LW} の生成と並行して，E_{LW} の各エッジの重みを *EdgeW* に保存する．*EdgeW*[e(C_1, C_2)] は E に含まれているエッジの内，C_1 に含まれているノードから C_2 に含まれているノードへ張られているエッジの数となるように計算する．

以上の手順において，E_{LW} の生成と *EdgeW* の計算に関して，図 6 の Web グラフを例に具体的に説明する．cluster()により，図 6(a)のノードの内 a と b が C_1 となり， d と e と f が C_2 という一つのクラスタになる．他のノードは当該ノードのみで構成される要素数 1 のクラスタとなる． V_{LW} が図 6(b)に示すようなかたちで得られた場合，同一クラスタ間に張られているエッジ，*e*(a, b), *e*(b, a), *e*(d, f), *e*(e, f), *e*(f, e)は，*e*(C_1, C_1), *EdgeW*[*e*(C_1, C_1)] = 2, *e*(C_2, C_2), *EdgeW*[*e*(C_2, C_2)] = 3 のように変換される．また，*e*(a, c), *e*(b, c)をクラスタ間エッジに変換すると，*e*(*C*[a], *C*[c]), *e*(*C*[b], *C*[c])となるが，*C*[a] = *C*[b] = C_1 であるため，これらはどちらも *e*(C_1, *C*[c])という同一のエッジである．このような場合，E_{LW} には *e*(C_1, *C*[c])を記録し，重複していたエッジ数が 2 なので *EdgeW*[*e*(C_1, *C*[c])] = 2 としてエッジの重みを記録する．他の E に含まれるエッジに関しても同様に処理していく．処理の結果は図 6(b)に示したようになる．図 6(b)中のエッジに添えてある数値は，*EdgeW* に記録されているエッジの重みである．



(a) **(b)**

図 6 縮小 Web の例

4. 評価実験

4.1. 実験データ

筆者らは e-society プロジェクト[11]においてクローリングを行っており，2006 年 9 月に収集された Web データには，日本語で記述されていると判断された Web ページが 147,010,114 ページ含まれている．本研究では，この日本語で記述されている Web ページのみで構成された Web リンク構造に対して，[12]で用いられている Host Level Reduction という手法を用い，ページ単位の Web リンク構造をホスト単位のリンク構造へと変換したものを実験データとして用いた．Host Level Reduction によって得られるホスト単位のリンク構造は，ページ単位のリンク構造から同ホスト間のリンクを除去し，異なるホスト間のリンクだけで構成されたグラフとなる．このデータは，ノード数 910,547，エッジ数 44,488,161 となるホスト単位のグラフとなっている．

4.2. Web グラフの縮小

3.2 項で提案した手法を実験データに適応することで得られる縮小 Web について，グラフ構造の圧縮率と，閾値 T_E, T_C との関係を図 7 に示す．なお，本実験では，3.2 項の comp()で用いる値αを$\alpha = 1.0\times10^{-15}$ とした．図 7 中に示す縮小 Web のうち，圧縮率によってどの程度の誤差が生じるかを調べるため，$T_E = 0.5$, $T_C = 0.3$ とした場合の縮小 Web Ⅰ，$T_E = 0.3$, $T_C = 0.7$ とした場合の縮小 Web Ⅱ，$T_E = 0.2$, $T_C = 0.2$ とした場合の縮小 WebⅢに対して 4.3 項で PageRank アルゴリズムを利用した評価を行う．縮小 Web Ⅰはエッジ圧縮率約 37.0%（エッジ数 16,472,867），ノード圧縮率約 89.3%（ノード数 813,089），縮小 Web Ⅱはエッジ圧縮率約 47.1%（エッジ数 20,955,590），ノード圧縮率約 89.7%（ノード数 816,977），縮小 WebⅢはエッジ圧縮率約 30.2%（エッジ数 13,439,150），ノード圧縮率約 83.6%（ノード数 761,092）に減少したものとなっている．

4.3. PageRank による評価

4.3.1. 縮小 Web に対する PageRank アルゴリズム

あるページ P_A から出ているリンクの数が N_A であるとする．PageRank アルゴリズムでは，取得したい PageRank の値を特に調節する必要がない場合，P_A を閲覧しているインターネットユーザが P_A のリンクを用いて他のページへ移動する確率(以降，推移確率)を，移動先ページに関わらず等しく $1/N_A$ としている．しかし，縮小グラフにおいて PageRank を計算する場合，インターネットユーザが，N_{Ci} 本のリンクを持つあるクラスタ C_i から C_j への推移確率を $1/N_{Ci}$ と置くと，本来クラスタ内のノード間に張られていたリンクの推移確率が無視されてしまうことになり，PageRank の計算に大きな誤差が出てくる恐れがある．そこで，縮小グラフにおける C_i から C_j への推移確率 P_{CiCj} は，3.2 項で述べた *EdgeW* を用いて，式(1)のように定義する．

$$P_{CiCj} = \frac{EdgeW\left[e\left(C_i, C_j\right)\right]}{N_{Ci}} \quad \cdots (1)$$

このようにエッジの重みを用いることで，PageRank アルゴリズムの反復計算において，本来クラスタ内部の各ノードに割り当てられる PageRank の値を，クラスタに近似的に集約することにつながる．そして，クラスタに集約された PageRank の値を，クラスタ内部のノードに分配することで，縮小 Web のもととなる Web グラフの PageRank を近似的に求めることができる．

以降，縮小 Web に対する PageRank の計算手順について数式とともに示す．縮小 Web $G_{LW}(V_{LW}, E_{LW})$において，グラフ内のクラスタ数を$|V_{LW}|$とし，PageRank を保存するための要素数 $|V_{LW}|$ の縦ベクトルとして *PR*, *prePR* を用いる．まず，$|V_{LW}|\times|V_{LW}|$の推移確率行列 **A** は，式(2)で定義される$|V_{LW}|\times|V_{LW}|$の行列 **B** を用いて式(3)によって得る．

$$\mathbf{B}[i][j] = \begin{cases} \dfrac{EdgeW\left(C_j, C_i\right)}{\sum_{e\left(C_j, C_k\right)\in E_{LW}} EdgeW\left(C_j, C_k\right)} & if\ \ {}^{\exists}e\left(C_j, C_i\right)\in E_{LW} \\ 0 & otherwise \end{cases} \quad \cdots (2)$$

$$\mathbf{A} = \frac{\alpha}{|V_{LW}|}\mathbf{1} + (1-\alpha)\mathbf{B} \quad \cdots (3)$$

ただし，式(3)中のαは PageRank の特許文書[13]に記されている，インターネットユーザがリンクに無関係に他のページへ移る確率であり，本稿では$\alpha = 0.15$ とした．また，式(3)中の **1** はすべての要素が 1 となる$|V_{LW}|\times|V_{LW}|$の行列である．次に，PR を式(4)に示すような形で初期化した後，縮小 Web のクラスタに割り振られる PageRank を式(5)の反復計算によって計

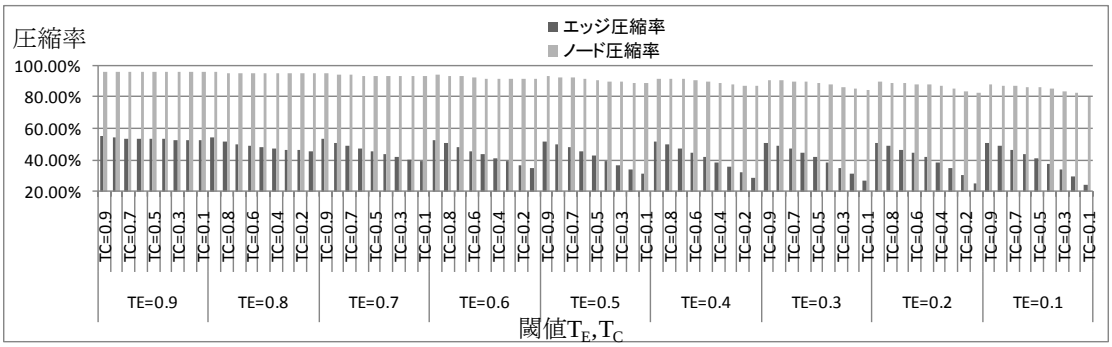


図 7 縮小 Web 構築における閾値 T_E, T_C と圧縮率の関係

算する.

$$PR[i] = \frac{1}{|V_{LW}|} \quad for\ \ all\ \ 1 \le i \le |V_{LW}| \qquad \cdots \quad (4)$$

$$\begin{aligned} &do\,\{ \\ &\quad prePR = PR; \\ &\quad PR = \mathbf{A} \cdot prePR; \\ &\}\ while\ (\|PR - prePR\|_1 < \varepsilon\ ); \end{aligned} \qquad \cdots \quad (5)$$

式(3)において, $\|\mathbf{A}\|_1$ は $\mathbf{A}$ の 1-ノルムであり, εは収束判定に用いる定数である. そして, 実際にもととなる Web グラフのノードの PageRank に対応する値は, クラスタの PageRank をクラスタに含まれるノード数で割った値になる. つまり, クラスタ C_iに含まれるノードの PageRank は, C_iに割り当てられている PageRank[i]を用いて, $PR[i]/|C_i|$として表わされる.

4.3.2. 縮小 Web の評価手法

評価においては, もととなった Web グラフと縮小 Web から得られる PageRank の比較, 縮小 Web における PageRank の実行時間の測定, もととなった Web グラフ・縮小 Web それぞれの PageRank によって得られるノードのランキングの比較の 3 点について評価を行った.

一つ目の評価方法として, PageRank アルゴリズムを利用する縮小 Web の評価として, もととなった Web グラフと縮小 Web それぞれに対して PageRank を計算し, 結果として得られる PageRank の値の比較を行う.

二つ目の評価方法として, 実際に縮小 Web を用いた場合の PageRank アルゴリズムの計算時間を測定し, もとになった Web グラフに対する PageRank アルゴリズムの計算時間と比較する.

三つ目の評価方法として, サンプリングしたノードの PageRank によるランキングの比較を行う. 具体的には, まず, グラフに含まれるノードの中からランダムに M 個をサンプリングし, ノード集合 P を用意する. そして, 計算した 2 つの PageRank をもとに, ノード集合 P をランキング順に整列させることで, PageRank 比較用のデータセットとなる, R_1, R_2 を作成する. R_1 をもととなった Web グラフから算出された PageRank によってノード集合 P を整列したものとし, R_2 は縮小 Web から算出された PageRank によってノード集合 P を整列したものとする. 評価は, この R_1, R_2 に対して, 吉田ら[14]によるランキング比較の手法を適用することで行う. ここで, R_1 の i 番目に位置するノードを $R_1[i]$と表記することとする. 吉田らの手法を R_1, R_2 に適用するに当たり, R_1 中の要素 $R_1[i]$それぞれに対し $w(R_1[i]) = M-i+1$ となるような重みを付ける. そして, 次のような関数 f を用いて

表 1 縮小 Web の PageRank 値誤差と PageRank 計算の実行時間

	無圧縮Web	縮小Web I	縮小Web II	縮小Web III
ノード圧縮率		89.3%	89.7%	83.6%
エッジ圧縮率		37.0%	47.1%	30.2%
similarity		0.880	0.902	0.824
PR中央値	2.43×10^{-7}	2.68×10^{-7}	2.61×10^{-7}	2.83×10^{-7}
PR平均値	1.10×10^{-6}	1.10×10^{-6}	1.10×10^{-6}	1.10×10^{-6}
PR標準偏差	2.04×10^{-5}	2.07×10^{-5}	1.93×10^{-5}	1.82×10^{-5}
PR誤差中央値		3.45×10^{-8}	2.87×10^{-8}	5.40×10^{-8}
PR誤差平均値		4.17×10^{-7}	3.71×10^{-7}	5.60×10^{-7}
PR誤差標準偏差		6.46×10^{-6}	6.94×10^{-6}	1.13×10^{-5}
PR計算時間	1m45s	54s	1m07s	44s

$$f(R_2[i]) = \begin{cases} w(R_2[i]) & if\ \ R_2[i] \in \{R_1[1], R_1[2], \cdots, R_1[L]\} \\ 0 & otherwise \end{cases} \qquad \cdots \quad (6)$$

R_1, R_2 の上位 L 個($0<L\le M$)の要素を

$$\begin{aligned} WR_1 &= [w(R_1[1]), w(R_1[2]), \cdots, w(R_1[L])] \\ WR_2 &= [f(R_2[1]), f(R_2[2]), \cdots, f(R_2[L])] \end{aligned} \qquad \cdots \quad (7)$$

という 2 つのベクトルを定義する(ただし, ノード $R_1[i]$と $R_2[j]$ が同一のノードである場合, $w(R_1[i]) = w(R_2[j])$であることに注意されたい). この 2 つのベクトルに対して次の式(8)

$$similarity = \frac{WR_1 \bullet WR_2}{|WR_1|\,|WR_2|} \qquad \cdots \quad (8)$$

を用いてコサイン類似度を計算し, 縮小 Web から算出された PageRank によるノードのランキングが, もととなった Web グラフの PageRank のよるランキングとどの程度類似しているかを評価する. この方法による評価では, (8)で得られる値 *similarity* が 1 に近ければ近いほど, 縮小 Web から算出される PageRank によるノードのランキングが正しく得られているとする.

4.3.3. 実験の結果と評価

4.3.2 で記したように, 縮小 Web の評価として, もととなった Web グラフと縮小 Web それぞれから算出した PageRank の値の比較と, 式(6)に記した *similarity* による評価の二つを行った. PageRank の値そのものの比較を行った結果および PageRank アルゴリズムの実行時間を表 1 に示す. 評価に使用した計算機のスペックは, CPU Intel Quad Core Xeon X5355 2.66GHz × 2, 物理メモリ 16GB である. 表 1 中の PR 中央値, PR 平均値, PR 標準偏差は, それぞれのもととなった Web

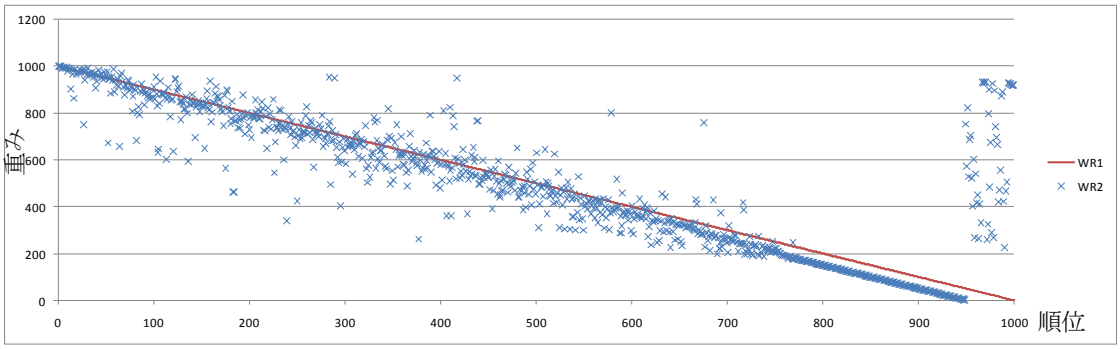


図 8 縮小グラフⅠのランキング誤差

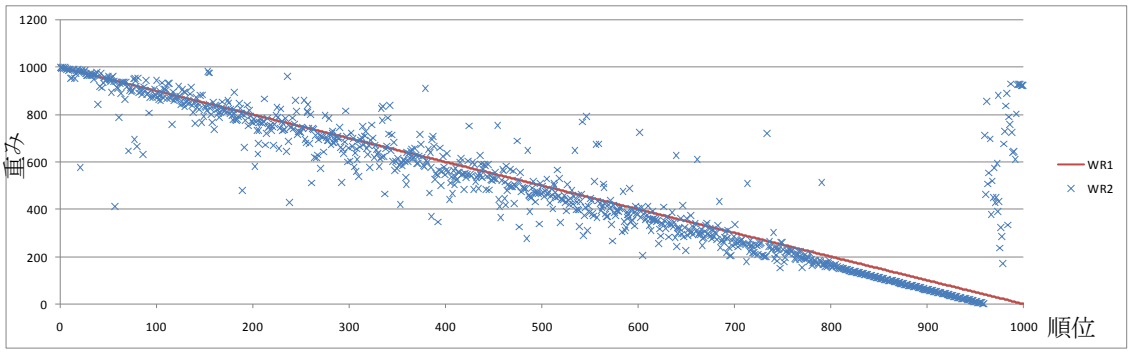


図 9 縮小グラフⅡのランキング誤差

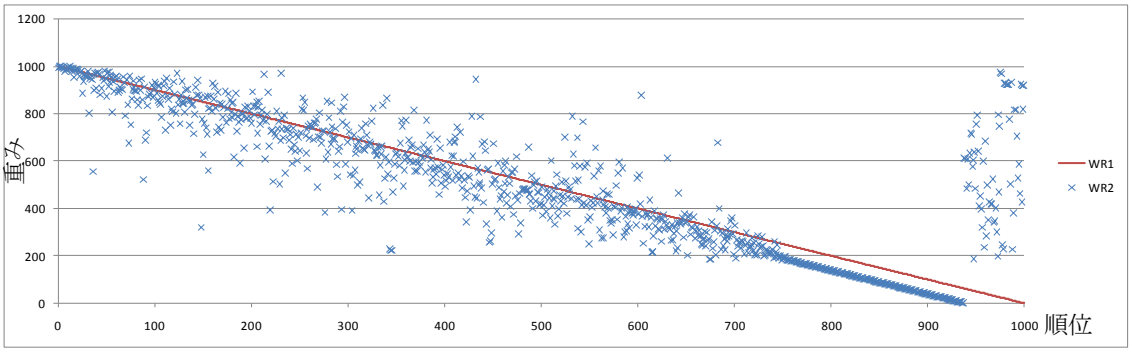


図 10 縮小グラフⅢのランキング誤差

グラフの PageRank を標本として計算した統計量である．また，PR 誤差中央値，PR 誤差平均値，PR 誤差標準偏差は，もととなった Web グラフと縮小 Web の PageRank に対し，ノードごとに PageRank の差の絶対値をとり，これを標本とした場合の統計量である．さらに，4.3.2 で *similarity* の計算に用いた縮小 Web Ⅰ～Ⅲの WR_1，WR_2 の一例を，横軸をランキング $x(1 \le x \le 1000)$, 縦軸を $w(R_1[x])$, $f(R_2[x])$ $(1 \le w(R_1[x]) \le 1000, 0 \le f(R_2[x]) \le 1000)$としてグラフに表すとそれぞれ，図 8～図 10 のようになる．グラフ上の点(1,1000)から(1000,1)に引かれた直線が WR_1 のグラフであり，この直線から WR_2 の値が離れているほど，ランキングの誤差が大きいことを示している．

4.3.4. 考察

図 8 と図 10 の散布図を比較すると，圧縮率の高い縮小 WebⅢの散布図である図 10 の方が，圧縮率の低い縮小 Web Ⅰの散布図である図 8 より WR_1 の直線から WR_2 のマーカー(散布図中の×印)が遠ざかっていることがわかる．同様に，表 1 に示した *similarity* も縮小 WebⅢの方が縮小 Web Ⅰより低い．これより，圧縮率の増加とともにランキングのずれも大きくなっていることがわかる．しかし，どちらの縮小 Web も 0.8 以上という高い *similarity* を維持したまま縮小することに成功している．ただし，図 8，図 9，図 10 ともに，グラフ右端(横軸 1000 付近)に大きくランキングが下がるノードが見られ，グラフの縮小によって局所的に大きな誤差が発生してしまっていることが分かる．この原因として，縮小 Web の PageRank 値を算出する際に，クラスタに集約された PageRank 値をクラスタ内のノードに均等に分配してしまっていることが考えられる．また，Authority 値と Hub 値によるノード間の類似度の評価では，リンクしているページとリンクされているページの集合が類似しているページを，近似的にしか判定できないことも原因の一つではないかと考えられる．

次に，ノードの圧縮率がほぼ等しくなっている縮小 Web Ⅰと縮小 WebⅡの関係について考察する．ノードの圧縮率に違いはないものの，縮小 Web Ⅰは縮小 WebⅡに比べて *similarity* が低く，表 1 中の PR 誤差平均値，PR 誤差中央値も大きい．縮小 Web Ⅰは縮小 WebⅡよりも T_C を低く設定して得られた縮小 Web であるので，縮小 Web Ⅰのクラスタは縮小 WebⅡのクラスタに比べてサイズが大きくなる傾向が

発生する．以上より，Web を縮小する場合，個々のクラスタサイズを極力抑えてクラスタリングを行った場合の方が PageRank のランキング制度は高くなることがうかがえる．つまり，本手法においては T_C を T_E よりも高く設定した方がよい結果を得られるということになる．

最後に，PageRank アルゴリズムを各縮小 Web に対して適用し，PageRank 値の計算にかかった実行時間について述べる．表 1 の PR 実行時間が PageRank 値の計算にかかった実行時間であるが，この計測時間は主に，隣接行列をメモリ上に読み込む時間および冪乗法の反復にかかる時間となっている．また，この実験で用いた PageRank アルゴリズム用プログラムは，隣接行列が疎行列であることを利用しているため，一回の冪乗法の反復にかかる計算コストのオーダがおおよそ $O(|V|+|E|)$ となっている．ただし，$|V|$ は Web グラフのノード数，$|E|$ は Web グラフのエッジ数である．実験の結果として，表 1 中でもっとも *similarity* の高い縮小 WebⅡを用いた場合でも，実行時間は 1 分 45 秒から 1 分 7 秒と，約 36%削減することに成功している．また，縮小 WebⅢでは，約 58%の時間短縮となっている．

5. おわりに

本稿では，HITS アルゴリズムのよって得られる Web ページの Authority 値と Hub 値をもとに Web リンク構造上の隣接ノード類似度を判定し，類似ノードクラスタリングすることで，グラフ構造を縮小化するという手法を提案した．実際に，クローリングによって収集した Web データに対して提案手法を適用し実験を行ったところ，4.3.2 で述べた PageRank 値によるランキング類似度の評価値 *similarity* を 0.9 に保った縮小 WebⅡを用いて，PageRank アルゴリズムの実行時間を 36%削減することに成功した．一方で，本手法の縮小 Web は PageRank 値の計算において局所的に誤差の大きくなるノードが発生するという性質があり，本来重要度の高いページがランキング上位から欠落してしまうという欠点を抱えている．今後の課題として，この局所的な誤差の改善とより圧縮率の高い縮小 Web を構築するための改善が挙げられる．また，PageRank アルゴリズム以外での縮小 Web の評価も今後の課題となる．

[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」及び科研(特定)「情報爆発に対応する高度にスケーラブルなモニタリングアーキテクチャ」によるものである．

[文献]

[1] J. Kleinberg, “Authority sources in a hyperlinked environment”, the 9th ACM-SIAM Symp. on Discrete Algorithms, pp.668-677, 1998.
[2] L. Page, S. Brin, R. Motowani and T. Winograd: “The PageRank Ciation Ranking: Bringing Order to the Web”, Tech. Rep., Stanford Digital Libraries, 1998.
[3] Taher H. Haveliwala, “Efficient computation of PageRank”, Tech. Rep., Computer Science Department, Stanford University, 1999.
[4] Y-Y Chen, Q. Gan, and T. Suel, “I/O-Efficient Techniques for Computing PageRank”, Proc. ACM Conf. on Information and Knowledge Management, pp.549-557, 2002.
[5] Sepandar D. Kamvar, Taher H. Haveliwala, C. Manning and G. Golub, “Adaptive methods for the computation of PageRank”, Tech. Rep., Stanford University, 2003.
[6] Sepandar D. Kamvar, Taher H. Haveliwala, C. Manning, and G. Golub, “Exploiting the block structure of the web for computating PageRank”, Tech. Rep., Stanford University, 2003.
[7] Sepandar D. Kamvar, Taher H. Haveliwala, C. Manning, and G. Golub, “Extrapolation methods for accelerating PageRank computation”, Proc. of 12th Int’l WWW Conf., Budapest, Hungary, pp.261-270, 2003.
[8] S. Gollapudi, M. Najork, and R. Panigraphy, “Using Bloom Filters to Speed Up HITS-Like Ranking Algorithms”, Proc. of 5th Int’l Workshop, WAW 2007, San Diego, CA, USA, December 2007.
[9] R. Lampel, and S. Moran, “The Stochastic Approach for Link-Structure Analysis(SALSA) and the TKC Effect”, Computer Networks and ISDN Systems, Vol.33, pp.387-401, 2000.
[10] G. Buehrer, and Kumar Chellapilla, “A Scalable Pattern Mining Approach to Web Graph Compression with Communities”, Proc. of Web Search and Data Mining 2008, WSDM2008, February 2008.
[11] e-society: http://cif.iis.u-tokyo.ac.jp/e-society/overview.html
[12] Yu Hirate and Hayato Yamana, “Web Structure in 2005”, Proc. of the 4th Workchop on Algorithms and Models for the Web-Graph(WAW2006), Splinger-Verlag, LNCS 4936, pp.36-46, 2008.
[13] L. Page, “METHOD FOR NODE RANKING IN A LINKED DATABASE”, United States Patent, Patent No. US6,285,999B1, September 2001.
[14] 吉田泰明，上田高徳，田代崇，平手勇宇，山名早人，”商用検索エンジンのランキングに関する定量的評価と特徴解析”, 情報研報（DBS）, Vol.2007, No.65, pp.441-446, July 2007.

片瀬 弘晶　Hiroaki KATASE
早大・基幹理工学研究科修士課程在学中．情報処理学会，DBSJ 各学生会員．
松永 拓　Taku MATSUNAGA
早大・基幹理工学研究科修士課程在学中．情報処理学会，DBSJ 各学生会員
上田 高徳　Takanori UEDA
早大・基幹理工学研究科修士課程在学中．ACM, IEEE，情報処理学会，電子情報通信学会，DBSJ 各学生会員．
田代 崇　Takashi TASHIRO
早大・基幹理工学研究科修士課程在学中．情報処理学会，DBSJ 各学生会員．
平手 勇宇　Yu HIRATE
2005 早大・理工学研究科修士課程修了．2008 早大・理工学研究科博士課程修了．博士（工学）．2006 年より同大・メディアネットワークセンター助手．ACM，IEEE，IPSJ，DBSJ 各会員．
山名 早人　Hayato YAMANA
1993 早大・理工学研究科博士課程了．博士（工学）．1993-2000 電総研．2000 早大・理工助教授．2005 同大理工学術院教授，現在に至る．IEEE，ACM，IEICE，IPSJ，DBSJ 各会員．