Random walk with restart に対する高速な Top-k 検索

藤原　靖宏[†,††]　　中辻　　真[†††]　　鬼塚　　真[†]　　喜連川　優[††]

† ＮＴＴサイバースペース研究所　〒 230–0847 神奈川県横須賀市光の丘 1-1
††† ＮＴＴサイバーソリューション研究所　〒 230–0847 神奈川県横須賀市光の丘 1-1
†† 東京大学生産技術研究所　〒 263–8505 東京都目黒区駒場 4-6-1
E-mail: †{fujiwara.yasuhiro,nakatsuji.makoto,onizuka.makoto}@lab.ntt.co.jp, ††kitsure@tkl.iis.u-tokyo.ac.jp

あらまし　グラフは基本的なデータ構造であり，世の中の様々なシステムで用いられている．グラフのノード間の類似度として近年 Random walk with restart (RWR) が提案され，様々なアプリケーションに応用されている．本論文ではグラフの中から RWR に基づいて問い合わせノードに対する類似ノードを K 個高速かつ正確に検索する問題を対象とする．提案手法では（１）特定のノードの類似度を疎行列を用いて計算する方法と，（２）不必要な類似度の計算を探索において省略する方法を用いる．実データを用いて比較実験を行い，提案手法は従来手法より高速に類似ノードを検索できることを確認した．

キーワード　グラフ，Random walk with restart，Top-k 検索

Efficient Top-k Search for Random Walk with Restart

Yasuhiro FUJIWARA[†,††], Makoto NAKATSUJI[†††], Makoto ONIZUKA[†], and Masaru KITSUREGAWA[††]

† NTT Cyber Space Laboratories, 1-1 Hikarinooka, Yokosuka, Kanagawa, 230–0847 Japan
††† NTT Cyber Solution Laboratories, 1-1 Hikarinooka, Yokosuka, Kanagawa, 230–0847 Japan
†† Institute of Industrial Science, The University of Tokyo, Komaba 4-6-1, Meguro, Tokyo 263–8505 Japan
E-mail: †{fujiwara.yasuhiro,nakatsuji.makoto,onizuka.makoto}@lab.ntt.co.jp, ††kitsure@tkl.iis.u-tokyo.ac.jp

1. はじめに

グラフはデータをノードとエッジで表現するデータ構造であり，様々な分野で用いられている．グラフ理論において２つのノード間の類似度は重要な性質の一つであり，ノード間の類似度を計算するために今まで様々な手法が提案されてきた [1], [2], [3]．その中でも Random walk with restart (RWR) はノード間の類似度を計算する手法として最も注目を集めているもののひとつである．RWR は今までグラフ理論でよく用いられてきたノード間の最短距離などを用いる手法と異なり，グラフの構造的な特徴に基づいて類似度が計算できるからである [4].

RWR は概念的に以下のように説明できる．問い合わせノード q からランダムウォークを開始し，隣接するノードにエッジの重みに比例した確率でランダムに移動する．さらにノードに到達するたびに一定の確率で問い合わせノード q に戻る．この操作を再帰的に繰り返した結果として各ノードにおける定常状態確率が得られるが，RWR はこの得られた定常状態確率を類似度とする方法である．すなわちグラフ上にあるノード u のノード q から見た類似度は，ノード q からランダムウォークをして得られたノード u の定常状態確率として計算される．

RWR は様々な分野のアプリケーションに応用されている計算方法であるが [5], [6], [7], [8]，計算コストが高いという問題がある．そのため今まで様々な高速化手法が提案されてきたが [8], [9]，それらは精度を犠牲にするものであった．しかしアプリケーションに対する応用を考えた場合，精度が犠牲になるのは好ましくない．また実際のアプリケーションにおいては問い合わせノード q からほかのすべてのノードの類似度を必ずしも計算するのではなく，類似度の高いノードの検索のみを行う処理が多く行われている [8]．そのため本論文では問い合わせノード q と検索個数 K が与えられたとき，問い合わせノード q に対して類似度の高いノードを K 個高速かつ正確に検索する問題に取り組む．提案手法では（１）特定のノードの類似度を疎行列を用いて計算する方法と，（２）類似度の低いノードの類似度計算を類似度の推定を行い省略する方法を用いる．提案手法と従来手法を実データを用いて比較した結果，提案手法は

比較手法より大幅に高速に検索を行えることを確認した．

本論文の構成は以下の通りである．2. 章で関連研究を述べる．3. 章で RWR の詳細な説明を行う．4. 章で提案手法の詳細を説明する．5. 章で実験結果を示す．6. 章で結論を述べる．

2. 関 連 研 究

RWR を用いたアプリケーションやその高速化手法について様々な研究が行われている．

自動画像キャプションは問い合わせ画像に対して自動的にそれを表現する言葉を割り当てる処理である．Pan らはグラフ構造を用いて問い合わせ画像に対して関連の高い言葉を割り当てる手法を提案した [5]．この手法は画像と言葉をノードとするグラフを作成し，RWR を用いて関連度の高い画像と言葉を計算する．彼らの手法は従来の手法に対して 10% 以上高い精度で自動画像キャプションを行えたことが報告されている．

レコメンデーションとはユーザに適した商品を提案する処理である．レコメンデーションにおける代表的な手法として協調フィルタリングがある [10]．この手法は推薦対象のユーザが過去に購入した商品に基づき，それらの商品を購入した別のユーザが購入した別の商品を推薦対象のユーザに推薦する．Konstas らはユーザとタグ，タグと商品にエッジを張ったグラフを作成し，RWR を用いてユーザと商品の関連度を計算し，関連度の高さに応じて推薦を行う手法を提案した [6]．Konstas らは実験を行い，彼らの手法が従来の協調フィルタリングより優れていることを示した．

Sun らは同じクラスタに属するノード間では RWR に基づく類似度が高くなることに着目し，RWR による類似度を高速に計算する手法を提案した [8]．彼らの手法はまずグラフをクラスタリングによって分割し，対象ノードのクラスタ内のノードのみに対して RWR のその他のクラスタ値を計算する．対象ノードのクラスタ外のノードの RWR の値は 0 とする．しかし彼らの手法では正確に類似度を計算できない．

Tong らは RWR の高速化手法として B_LIN とその派生である NB_LIN を提案した [9]．彼らの手法はグラフをクラスタリングし，行列近似を用いて RWR の値を近似するものである．特に彼らは NB_LIN に対して特異値分解を行列近似に用いることにより，RWR の値の近似誤差の理論値を求められることを示した．そして実験において Sun らの手法より彼らの手法がより高速に RWR の値を求められることを示した．しかし Tong らの手法は $O(n^2)$ の計算コストを要する．これは彼らの手法は類似度を求めるために $O(n^2)$ の大きさの行列の計算を行うからである．

3. Random walk with restart

この章では RWR の説明をする．表 1 に記号とその定義を示す．

RWR はグラフのノードの類似度を計算するひとつの方法である．RWR では問い合わせノード q を始点とするランダムウォークを行う [4]．そしてランダムウォークでノードに到達するたびに一定の確率 c で問い合わせノードに戻る．$\mathbf{p}$ を $n\times1$ 行列とし，要素 p_u をノード u にランダムウォークする確率を表すとする．また $\mathbf{q}$ を $n\times1$ 行列とし，要素 q_q を 1，その他の要素を 0 とする．また $\mathbf{A}$ を列が正規化されたグラフの隣接行列とする（すなわち要素 $A_{u,v}$ はノード u からノード v へランダムウォークする確率を表す）．定常状態における各ノードにおける存在確率は以下の式を再帰的に収束するまで繰り返すことで計算することができる．

表 1　主な記号の定義

記号	定義
q	問い合わせノード
K	解ノードの数
n	グラフにおけるノードの数
m	グラフにおけるエッジの数
c	問い合わせノードに戻る確率
$\mathbf{p}$	ノード u の類似度となる $n\times1$ 行列
$\mathbf{q}$	q 番目の要素が 1 でその他の要素が 0 となる $n\times1$ 行列
$\mathbf{A}$	列が正規化されたグラフの隣接行列

$$\mathbf{p} = (1-c)\mathbf{A}\mathbf{p} + c\mathbf{q} \tag{1}$$

定常状態における確率は問い合わせノードを中心に高くなるが，RWR はこの確率を類似度とする方法である．すなわち行列 $\mathbf{p}$ の要素 p_u はノード u のノード q に対する類似度となる．

定義から繰り返し回数を t としたとき RWR の計算コストは $O(mt)$ となる．そのためグラフが大規模である場合 RWR の値を計算するのは非常に時間がかかる．そのため高速に RWR の値を計算する手法が求められている [6]．

4. 提 案 手 法

この章では提案手法の詳細について述べる．まず 4.1 章にて提案手法の概要について述べ，4.2 章と 4.3 章にて提案手法で用いる 2 つの手法を説明する．そして 4.4 章にて検索アルゴリズムについて説明する．なおこの章においてすべての補助定理の証明は紙幅の関係から省略する．

4.1 手 法 概 要

提案手法は以下の 2 つの手法から構成される．

疎行列計算

3. 章で述べたとおり問い合わせノードに対するグラフ上の各ノードの類似度は定常状態における確率として求めることができる．この方法はグラフのすべてのノードの類似度を計算するため計算コストが高い．そこで提案手法ではすべてのノードの類似度を計算せずに，特定のノードの類似度のみを計算し高速な検索を可能にする．

特定のノードの類似度は式 (1) から直接求められる逆行列を用いることで計算できる．そのためこの逆行列を事前に計算しておけば特定のノードの類似度を計算できる．しかし逆行列を保持するには一般的に $O(n^2)$ のメモリが必要になるのが問題になる．

そこで提案手法ではこの逆行列を疎行列として保持するために，検索の事前処理でまずノードを並び替えてから LU 分解を計算し，得られた上三角行列・下三角行列の逆行列を計算する．この上三角行列・下三角行列の逆行列は疎行列なため，隣接リ

スト表現 [11] を用いることにより特定のノードの類似度を疎行列から計算できる．結果として特定のノードの類似度を少ないメモリ量で高速に計算することができる．なおグラフが疎な構造でない場合であっても，この手法を用いればエッジの数の計算量で類似度を計算できる．

木構造による推定

上記の疎行列計算の手法により特定のノードの類似度を高速に計算することができる．提案する 2 つめの手法では K 個のノードの検索においてどのノードの類似度を計算し，どのノードの類似度の計算を枝狩りするべきかを決めるためにノードの類似度を推定する．結果的に類似度を計算しないノードを枝刈りでき，検索を高速に行うことができる．

ノードの類似度を推定するために提案手法では RWR による類似値が問い合わせノードのからホップ数が増えるほど小さくなること，またノードの類似度は計算済みの類似度から推定できることを用いる．提案手法ではまず問い合わせノードから幅優先探索を行い，ホップ数の小さい順にノードの類似度の推定値を計算する．そしてそのノードが解になり得る場合，疎行列から類似度を計算する．類似度の推定値はすでに求めた類似度から計算するが，ホップ数の小さい順にノードを調べることにより，より効果的に推定を行うことができる．ノードの推定値は $O(1)$ で計算することができるため，少ない計算コストで類似度の計算を省略することができる．

4.2 疎行列計算

この章では特定のノードの類似度は逆行列から計算できることを述べ，その逆行列を疎行列から計算する手法について述べる．

4.2.1 類似度計算

式 (1) から以下のように類似度を計算できる．

$$\mathbf{p} = c\{\mathbf{I} - (1-c)\mathbf{A}\}^{-1}\mathbf{q} = c\mathbf{W}^{-1}\mathbf{q} \tag{2}$$

ここで $\mathbf{I}$ は単位行列であり，$\mathbf{W} = \mathbf{I} - (1-c)\mathbf{A}$ である．この式から特定のノードの類似度は逆行列 $\mathbf{W}^{-1}$ の対応する要素を用いることで計算できることがわかる．しかし一般的に行列 $\mathbf{W}$ が疎であってもその逆行列は密になるため [12]，直接逆行列を用いる手法は多くのメモリ量が必要になる．

そこで提案手法ではノードを並び替えて上三角行列と下三角行列の逆行列を疎な行列として保持し，逆行列を計算する．定式的には行列 $\mathbf{W}$ を LU 分解し $\mathbf{W} = \mathbf{LU}$ とすることで問い合わせノードの類似度を以下のように計算する．

$$\mathbf{p} = c\mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{q} \tag{3}$$

ここで $\mathbf{L}^{-1}$ と $\mathbf{U}^{-1}$ はそれぞれ下三角行列と上三角行列になる．

具体的なノードの並び替え方法を述べる前に行列 $\mathbf{L}^{-1}$ と $\mathbf{U}^{-1}$ の要素は行列 $\mathbf{L}$ と $\mathbf{U}$ の要素を用いて計算できることを示す．前進後退代入を用いることにより行列 $\mathbf{L}^{-1}$ と $\mathbf{U}^{-1}$ の要素は以下のように計算できる [12].

$$L_{ij}^{-1} = \begin{cases} 0 & (i < j) \\ 1/L_{ij} & (i = j) \\ -1/L_{ii} \sum_{k=j}^{i-1} L_{ik} L_{kj}^{-1} & (i > j) \end{cases} \tag{4}$$

$$U_{ij}^{-1} = \begin{cases} 0 & (i > j) \\ 1/U_{ij} & (i = j) \\ -1/U_{ii} \sum_{k=i+1}^{j} U_{ik} U_{kj}^{-1} & (i < j) \end{cases} \tag{5}$$

またクラウト法を用いることにより行列 $\mathbf{L}$ と $\mathbf{U}$ の要素は行列 $\mathbf{W}$ を用いて以下のように計算できる [12].

$$L_{ij} = \begin{cases} 0 & (i < j) \\ 1 & (i = j) \\ 1/U_{jj} \left(W_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} \right) & (i > j) \end{cases} \tag{6}$$

$$U_{ij} = \begin{cases} 0 & (i > j) \\ W_{ij} & (i \leqq j \cap i = 1) \\ W_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj} & (i \leqq j \cap i \neq 1) \end{cases} \tag{7}$$

式 (4), (5), (6), (7) は行列 $\mathbf{L}^{-1}$, $\mathbf{U}^{-1}$, $\mathbf{L}$, $\mathbf{U}$ の要素がそれぞれ列を左から右の要素の順に，また同じ列は上から下の要素の順に計算できることがわかる．例えば行列 L_{ij}^{-1} の要素は行列 $\mathbf{L}$ と $\mathbf{L}^{-1}$ の対応する上と左の要素から計算でき，行列 L_{ij} の要素は行列 $\mathbf{W}$, $\mathbf{L}$, $\mathbf{U}$ の対応する上と左の要素から計算できる．

提案手法は上三角行列と下三角行列の逆行列についての以下の 3 つの考察に基づいている．(1) 行列 L_{ij}^{-1} と U_{ij}^{-1} は対応する行列 $\mathbf{L}$ と $\mathbf{U}$ の上または左の要素が 0 であれば 0 になる．(2) 行列 $\mathbf{L}$ と $\mathbf{U}$ の上または左の要素は行列 $\mathbf{W}$ の上または左の要素が 0 であれば 0 になる．(3) $\mathbf{W}$ の上または左の要素は行列 $\mathbf{A}$ の上または左の要素が 0 であれば 0 になる．すなわち行列 $\mathbf{A}$ の上または左の要素が 0 になるようにすれば行列 L_{ij}^{-1} と U_{ij}^{-1} を疎にすることができる．

この考察に基づき上三角行列と下三角行列の逆行列を疎にするための手法について以下の 3 つのものを提案する．

次数による並び替え

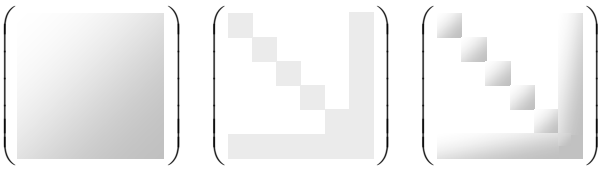
この方法ではグラフのノードをその次数が小さい順に並び替える．次数が小さいノードは少ないエッジでほかのノードとつながっているため，この並び替えに対応する行列 $\mathbf{A}$ の左と上の要素は 0 となる．

クラスタリングによる並び替え

この方法では Newman clustering [13] によってグラフを κ 個のクラスタに分割しその結果からノードを並び替える．クラスタの数は Newman clustering により自動的に決定される．クラスタに分割してから空の κ+1 番めのクラスタを新たに作り，1 番目から κ 番目のクラスタにおいてクラスタをまたがるエッジをノードが持つ場合，そのノードを κ+1 番目のクラスタに移動する．その結果，行列 $\mathbf{A}$ において 1 番目から κ 番目のクラスタのノードはすべてクラスタ内にしかエッジを張ってなく， κ+1 番目のクラスタのノードのみ複数のクラスタにエッジを張っている形になる．

併用による並び替え

この方法では次数による並び替えとクラスタリングによる並び替えを併用する．すなわちまずクラスタリングによる並び替



(1) 次数　　(2) クラスタリング　　(3) 併用

図 1　ノードの並び替え方法.

えによってノードを並び替えてから，それぞれのクラスタ内において次数の小さい順にノードを並び替える．

図 1 に上の 3 つの方法で得られる行列 **A** を示す．ここで値が 0 の要素は白，そうでない要素は灰色で表現する．

これらの手法によって上三角行列と下三角行列の逆行列は疎になり，隣接リスト表現 [11] を用いることにより少ないメモリ量で逆行列を計算できる．

4.3　木構造による推定

この章では検索の途中において類似度を計算していないノードの類似度を高速かつ効果的に推定する方法について述べる．この方法は幅優先探索木の構造を利用して類似度の上限値を計算する．そのため検索途中の暫定的な解のノードの類似度より推定値が小さい場合，類似度の計算を省略することができる．この章ではまず推定において用いる記号について説明してから，推定式の定義を述べる．そして最後に推定式は検索の途中において逐次的に更新できることを述べる．

4.3.1　推定において用いる記号

検索ではまず問い合わせノードを始点にする幅優先探索木を構築する．その結果，幅優先探索木の第 0 層は問い合わせノードとなり，第 1 層は問い合わせノードに直接つながっているノードとなり，第 i 層は問い合わせノードからのホップ数が i となるノードとなる．

グラフにおけるノードの集合を V とし，類似度を計算したノードを V_s とする．ノード u の層番号を l_u とする．さらに層番号が l_u でありかつ類似度を計算したノードの集合を $V(l_u)$ とする．すなわち $V(l_u) = \{v : (v \in V_s) \cap (l_v = l_u)\}$ である．隣接行列 **A** において最も値の大きいエッジの重みを A_{max}，すなわち $A_{max} = \max\{A_{ij} : i, j \in V\}$ とする．またノード u から出ている最も値の大きいエッジの重みを $A_{max}(u)$，すなわち $A_{max}(u) = \max\{A_{iu} : i \in V\}$ とする．なお A_{max} と $A_{max}(u)$ は検索を行う前に計算できる．

4.3.2　類似度の推定

ここではノードの類似度の推定値の定義と，推定値が類似度より小さくなることがないことを示す．ノード u の推定値 $\bar{p}_u$ は幅優先探索木の構造を用いて定義する．

[**定義** 1]（類似度の推定） 問い合わせノード q でないノード u の推定値 $\bar{p}_u$ は以下の式から計算する．

$$\bar{p}_u = c' \left\{ \sum_{v \in V(l_u - 1)} p_v A_{max}(v) + \sum_{v \in V(l_u)} p_v A_{max}(v) + \left(1 - \sum_{v \in V_s} p_v\right) A_{max} \right\} \tag{8}$$

ここで $c' = (1 - c)/(1 - A_{uu} + cA_{uu})$ である．またノード u が問い合わせノードであるとき $\bar{p}_u = 1$ とする．

定義 1 を用いて推定値を計算するには $O(n)$ の計算コストが必要である．これは集合 $V(l_u - 1)$ と $V(l_u)$ と V_s の大きさが $O(n)$ だからである．この推定値を $O(1)$ の計算コストで計算するためのアプローチは 4.3.3 章で述べる．

この推定値の性質を示すため，以下の補助定理を導入する．

[**補助定理** 1]（類似度の推定） ノード u に対して $\bar{p}_u \geqq p_u$ が成り立つ．

検索においては問い合わせノードを始点とする幅優先探索木を構築するが，これは問い合わせノードからホップ数が小さいほど類似度の値は大きくなり，また推定は計算した類似度から行うため，より効果的に推定を行えるためである．

検索では幅優先探索の順番にノードの推定値を計算し，それが解候補のノードの最も小さい類似度より小さい場合，検索を打ち切る．その他の類似度を計算せずに検索を打ち切っても正しい検索結果になることを示すために，この推定についての以下の性質を示す．

[**補助定理** 2]（木構造の探索） 幅優先探索の順番にノードを検索していれば，$l_u \leqq l_v$ となるノード u と v に対して $\bar{p}_u \geqq \bar{p}_v$ が成り立つ．

補助定理 2 からあるノードにおける推定値はそのノードと同じかまたは下の層にあるノードの推定値より小さくならないことがわかる．そのため類似度を計算していないノードの推定値が解候補のノードの最も小さい類似度より小さい場合，検索を打ち切っても検索結果に影響は出ない．

4.3.3　逐次的計算

4.3.2 章で述べたとおり，定義 1 を用いてそれぞれのノードの推定値を計算するには $O(n)$ の計算コストが必要である．ノードの推定値を高速に計算する手法について述べる．この章においてノード u はノード u' の直後に類似度を計算されるとする．すなわち検索ではノード u'，ノード u の順番で類似度を計算するとする．また $\bar{p}_{u,1}$ と $\bar{p}_{u,2}$ と $\bar{p}_{u,3}$ はそれぞれ定義式 (8) における第一項，第二項，第三項とする．すなわち $\bar{p}_u = c'(\bar{p}_{u,1} + \bar{p}_{u,2} + \bar{p}_{u,3})$ となる．

検索においてノード u の推定値を以下のように計算する．

[**定義** 2]（逐次的計算） もしノード u が問い合わせノードでないとき，推定式の第一項，第二項，第三項をそれぞれ以下のように計算する．

$$\bar{p}_{u,1} = \begin{cases} \bar{p}_{u',1} & \text{if } l(u) = l(u') \\ \bar{p}_{u',2} + p_{u'} A_{max}(u') & \text{otherwise} \end{cases}$$

$$\bar{p}_{u,2} = \begin{cases} \bar{p}_{u',2} + p_{u'} A_{max}(u') & \text{if } l(u) = l(u') \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\bar{p}_{u,3} = (\bar{p}_{u',3}/A_{max} - p_{u'})\, A_{max}$$

もしノード u が問い合わせノードであるとき，$\bar{p}_{u,1} = p_q A_{max}(q)$，$\bar{p}_{u,2} = 0$，$\bar{p}_{u,3} = (1 - p_q) A_{max}(u)$ とする．

上記の計算式について以下の補助定理を示す．

[**補助定理** 3]（逐次的計算） 定義 2 を用いれば，ノード u の

Algorithm 1 類似検索

Input: q, 問い合わせノード
K, 類似ノードの数
$\mathbf{L}^{-1}$, $\mathbf{L}$ の逆行列
$\mathbf{U}^{-1}$, $\mathbf{U}$ の逆行列
Output: V_a, 類似ノードの集合

```
 1: θ = 0;
 2: V_s = ∅;
 3: V_a = ∅;
 4: K 個のダミーノードを V_a に加える;
 5: ノード q から幅優先探索木を計算;
 6: while V_s ≠ V do
 7:    u := argmin(l_v | v ∈ V\V_s);
 8:    ノード u の類似度の推定値 p̄_u を計算;
 9:    if p̄_u < θ then
10:       return V_a;
11:    else
12:       L^{-1} と U^{-1} を用いて類似度 p_u を計算;
13:       if p_u > θ then
14:          v := argmin(p_w | w ∈ V_a);
15:          ノード v を V_a から取り除く;
16:          ノード u を V_a に加える;
17:          θ := min(p_w | w ∈ V_a);
18:       end if
19:    end if
20:    ノード u を V_s に加える;
21: end while
22: return V_a;
```

RWR による類似度の推定値を正確に $O(1)$ の計算コストで計算できる．

4.4 探索アルゴリズム

アルゴリズム 1 に問い合わせノードに対して K 個の類似度の高いノードを検索するアルゴリズムを示す．ここで θ を解候補のノードの最小の類似度とし，V_a を解候補のノードとする．

検索ではまず K 個のダミーのノードを解候補のノードとする（4行目）．ダミーのノードの類似度はすべて 0 とする．そして幅優先探索木を構築する（5行目）．層番号の順にノードを選択し（7行目），選ばれたノードの推定値を計算する（8行目）．もし推定値が θ より小さい場合，そのノードは解になり得ず（補助定理 1），またその他の選択されなかったノードの推定値も θ より小さくなる（補助定理 2）．そのため検索を終了する（9，１０行目）．もしそうでなければ選択されたノードが解になり得る．そのためそのノードの類似度を計算する（１２行目）．もし計算した類似度が θ より大きい場合，V_a と θ を更新する（１３〜１８行目）．

5. 評価実験

提案手法の性能を調べるために Tong らによって提案された NB_LIN [9] と比較実験を行った．なおここで提案手法とは２つの手法（疎行列計算と木構造による推定）を用いたものである．NB_LIN は彼らが報告しているとおり 3. 章で述べた繰り返しによる手法や Sun らによって提案された手法 [8] より高速であり，また同じく Tong らによって提案された B_LIN とほぼ同じ性能であることが知られている．実験では Tong らが提案しているように近似手法として特異値分解を用いた．実験では過去の研究と同様に問い合わせノードに戻る確率 c を 0.95 とした [9], [14]．実験では以下のデータを用いた．

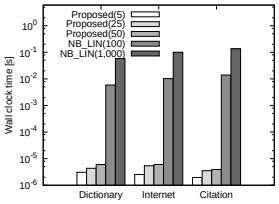


図 2 検索時間

- *Dictionary* (注1): このデータはオンライン辞書の FOLDOC で得られる単語のネットワークである(注2)．このデータでは単語 u の説明に単語 v が使われているときにノード u とノード v にエッジを張っている．ノードの数は 13,356 でありエッジの数は 120,238 である．
- *Internet* (注3): このデータはインターネットの構造をグラフ構造として表現したものである．このデータは Oregon Route Views Project (注4) より提供されているものであり，BGP テーブルを解析して得られたものである．ノードの数は 22,963 でありエッジの数は 48,436 である．
- *Citation* (注5): このデータは論文の共著関係のグラフである．論文は Condensed Matter E-Print (注6) に投稿されたものを対象にしている．ノードの数は 31,163 でありエッジの数は 120,029 である．

この章ではノードの並び替え方法として併用による並び替えを用いて，上位５個のノードを検索した結果を示す．

実験は CPU が Intel Xeon Quad-Core 3.33GHz，メモリが 32GB の Linux サーバで行った．またすべてのアルゴリズムは GCC で実装した．

5.1 高速性

提案手法と NB_LIN の検索時間を比較した．図 2 に結果を示す．提案手法は K の値によって検索時間が異なるため，この図において *Propased(K)* として提案手法の検索時間を示す．また NB_LIN は近似を行う特異値の数により検索時間が異なるため，特異値の数を 100 としてその結果を *NB_LIN(100)*，特異値の数を 1,000 としてその結果を *NB_LIN(1,000)* と示す．

この図から提案手法は NB_LIN より大幅に高速であることがわかる．これは K 個の類似ノードを検索するために NB_LIN はすべてのノードの類似度を計算しなければならないが，提案手法は類似度の低いノードを枝狩りしている結果，類似度の計算回数が少ないためである．

5.2 検索の精度

提案手法の優位性の一つとして検出結果が正確であることがあげられる．従来手法は検出結果が正確でないにしろ，どの程度正確に検出を行うことができるのかを示すことは提案手法の優位性を検証するために必要である．そのために比較実験を行った．

(注1)：http://vlado.fmf.uni-lj.si/pub/networks/data/dic/foldoc/foldoc.zip
(注2)：http://foldoc.org/
(注3)：http://www-personal.umich.edu/ mejn/netdata/as-22july06.zip
(注4)：http://routeviews.org/
(注5)：http://www-personal.umich.edu/ mejn/netdata/cond-mat-2003.zip
(注6)：http://arxiv.org/archive/cond-mat

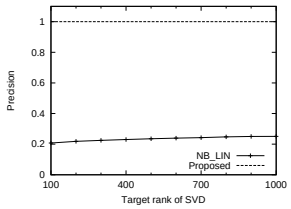


図 3　特異値の数と正解率の関係

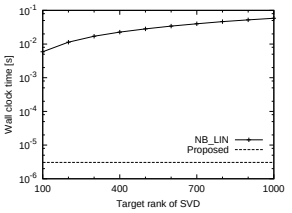


図 4　特異値の数と検索時間の関係

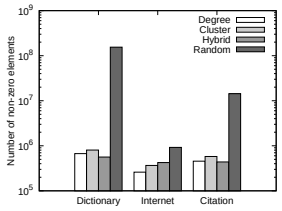


図 5　ノードの並び替えの効果

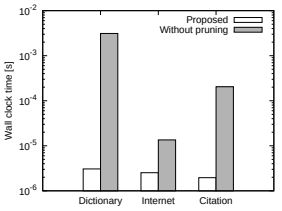


図 6　木構造による推定の効果

解の正確性を評価する指標として正解率を用いた．正解率は検索結果の類似ノードが，繰り返しによる手法で得られる類似ノードにも含まれる割合である．図 3 と図 4 にそれぞれ提案手法と NB_LIN の正解率と検索時間を示す．NB_LIN においては特異値の値を変えて実験を行った．実験データとしては Dictionary を用いた．

図 3 からわかるとおり，提案手法の正解率は 1 である．これは提案手法が正確に探索を行えるからである．一方 NB_LIN の正解率は提案手法より低いことがわかる．さらに図 4 から NB_LIN には検索時間と正解率にトレードオフがあることがわかる．すなわち NB_LIN は特異値の数が多くなるほど正解率は上昇するが検索時間は増加してしまう．

5.3　手法の有効性

以下の実験では提案手法において用いている 2 つのアプローチについて検証を行う．

5.3.1　ノードの並び替え

提案手法では類似度を計算するために上三角行列と下三角行列の逆行列を用いるが，これらの行列を疎にするためにノードの並び替えを行う．ノードの並び替えとして 3 つのアプローチを提案したが，これらが逆行列を疎にするためにどの程度効果的であるかの検証を行った．図 5 に各手法による逆行列における非ゼロ要素の数を示す．この図において *Degree* とは次数による並び替え，*Clustering* とはクラスタリングによる並び替え，*Hybrid* とは併用による並び替え，*Random* とはノードをランダムに並び替えた結果を示している．

この図から提案した 3 つの方法ともランダムにノードを並び替えるより逆行列を疎にするために有効であることがわかる．また非ゼロ要素の数はグラフにおけるエッジの数に比例していることがわかる．すなわち提案手法におけるメモリ量は $O(m)$ となる．

5.3.2　類似度の推定

4.3 章で述べたとおり，提案手法では類似度を推定し必要のない類似度の計算を省略する．この方法の有効性を示すために，提案手法においてこの方法を用いないものと比較を行った．実験結果を図 6 に示す．この図において提案手法から類似度を推定し枝狩りしない方法の実験結果を *Without pruning* として示す．

提案手法は類似度を推定しない方法より 1,020 倍高速であった．提案手法は一つ一つノードを選択し類似度の推定を行い，もし推定値が解候補のノードにおける最小の類似度より小さければ検索を打ち切る．そのため類似度を推定すると高速に検索が終了でき，結果大幅な高速化が可能になる．

5.4　ケーススタディ

ここでは提案手法と NB_LIN では検索の結果が大きく異なることの実例を示す．実験では 3 つの OS の名前に対してデータとして Dictionary を用いて検索を行った．NB_LIN において特異値の数は 1,000 とした．結果を表 2 に示す．

提案手法の結果は直感的に類似単語として妥当なものが得られる結果となった．例えば Microsoft Windows に対する類似単語の結果は Microsoft Windows，W2K，Windows/386，Windows 3.0，Windows 3.11 となった．検索結果はすべて Microsoft の OS の名前であり， Microsoft の OS 市場における優位性を反映した結果になった．

一方 Mac OS の類似単語の結果にはいくつかの Apple 独自の技術が含まれる．例えば Macintosh user interface は Apple の PC に使われている GUI の名称である．また Macintosh file system は Mac OS にのみ用いられているファイルシステムのことである．

Linux についての検索結果にはその開発に関係する単語を検索できた．Linux の開発には多くのプロジェクトが関わっているが，検索結果に出てきた Linux Documentation Project はその中のひとつである．

しかし NB_LIN の検索結果は提案手法と異なり，提案手法の検索結果の方がより直感的な検索結果となった．

6.　ま　と　め

この論文では RWR に基づき類似ノードを高速に検索する問題について取り組んだ．提案手法は（1）特定ノードの類似度を逆行列を用いて高速に計算する方法と，（2）検索に不必要な類似度の計算を省略する方法を用いた．提案手法と従来手法を比較したところ，検索の精度を犠牲にすることなく提案手法は従来手法より高速に検索が行えることを確認した．

文　　献

[1] Y. Koren, S. C. North and C. Volinsky: “Measuring and extracting proximity in networks”, KDD, pp. 245–255 (2006).
[2] H. Tong, C. Faloutsos and Y. Koren: “Fast direction-aware proximity for graph mining”, KDD, pp. 747–756 (2007).
[3] D. Lizorkin, P. Velikhov, M. N. Grinev and D. Turdakov: “Accuracy estimate and optimization techniques for simrank computation”, PVLDB, **1**, 1, pp. 422–433 (2008).
[4] H. Tong and C. Faloutsos: “Center-piece subgraphs: problem definition and fast solutions”, KDD, pp. 404–413 (2006).
[5] J.-Y. Pan, H.-J. Yang, C. Faloutsos and P. Duygulu: “Automatic multimedia cross-modal correlation discovery”, KDD, pp. 653–658 (2004).
[6] I. Konstas, V. Stathopoulos and J. M. Jose: “On social networks and collaborative recommendation”, SIGIR, pp. 195–202 (2009).

表 2　提案手法と NB_LIN による ‘Microsoft Windows’, ‘Mac OS’, ‘Linux’ に対する検索結果.

検索単語	検索手法	ランキング				
		1	2	3	4	5
Microsoft Windows	Proposed	Microsoft Windows	W2K	Windows/386	Windows 3.0	Windows 3.11
	NB_LIN	Microsoft Windows	Microsoft Networking	Microsoft Network	W2K	Thumb
Mac OS	Proposed	Mac OS	Macintosh user interface	Macintosh file system	multitasking	Macintosh Operating System
	NB_LIN	Mac OS	Rhapsody	SORCERER	Macintosh Operating System	PowerOpen Association
Linux	Proposed	Linux	Linux Documentation Project	Unix	lint	Linux Network Administrators' Guide
	NB_LIN	Linux	Linux Documentation Project	SL5	debianize	SLANG

[7] D. Liben-Nowell and J. M. Kleinberg: “The link prediction problem for social networks”, CIKM, pp. 556–559 (2003).

[8] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos: “Neighborhood formation and anomaly detection in bipartite graphs”, ICDM, pp. 418–425 (2005).

[9] H. Tong, C. Faloutsos and J.-Y. Pan: “Fast random walk with restart and its applications”, ICDM, pp. 613–622 (2006).

[10] J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl: “An algorithmic framework for performing collaborative filtering”, SIGIR, pp. 230–237 (1999).

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein: “Introduction to Algorithms”, The MIT Press (2009).

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery: “Numerical Recipes 3rd Edition”, Cambridge University Press (2007).

[13] A. Clauset, M. E. J. Newman and C. Moore: “Finding community structure in very large networks”, Physical Review E, pp. 1– 6 (2004).

[14] J. He, M. Li, H. Zhang, H. Tong and C. Zhang: “Manifold-ranking based image retrieval”, ACM Multimedia, pp. 9–16 (2004).