Personalized PageRank に対するアドホックな検索手法

藤原　靖宏[†a)]　中辻　真[††]　塩川　浩昭[†]　三島　健[†]
鬼塚　真[†,†††]

Fast Ad-Hoc Search Algorithm for Personalized PageRank

Yasuhiro FUJIWARA[†a)], Makoto NAKATSUJI[††], Hiroaki SHIOKAWA[†], Takeshi MISHIMA[†], and Makoto ONIZUKA[†,†††]

あらまし　*Personalized PageRank (PPR)* はグラフにおけるノード間の代表的な類似度であり，PPR に対するノードの検索は多くのアプリケーションにおいて用いられている．多くのアプリケーションにおいてグラフは動的に変化する性質があるなどの理由から，PPR に対してアドホックな検索が行えることが望ましい．アドホックな検索とは，検索を行うごとに検索対象のグラフやパラメータを変えながら検索を行うことである．しかしグラフのサイズが大きくなるとアドホックに検索を行う計算コストが高くなるという問題がある．過去にも PPR の計算コストを低減するためにさまざまな手法が提案されてきた．しかしそれらの手法は事前計算が必要であったり，近似的な検索結果になるなどの理由から有効にアドホックな検索を行えないという問題があった．本論文では PPR に対してアドホックな検索を行う高速化手法 *Castanet* を提案する．提案手法は (1) 再帰的に PPR のスコアの下限値と上限値を推定し，(2) 繰り返し計算の中で動的に検索結果を得るのに不必要なノードを枝狩りすることを特徴とする．実験から提案手法は既存手法より高速に PPR に対してアドホックな検索が可能であること確認した．

キーワード　Personalized PageRank，アドホック，高速，アルゴリズム

1. まえがき

近年ソーシャルネットワークに代表される大規模なグラフのマイニングを行うことへの関心が高まっている．PageRank [1] が発表されて以降データベースの分野においてグラフのリンク構造に基づいたノードの類似度についての研究が進んでいる．それはノードの類似度を用いることによりグラフデータベースにおいてグラフを分類や解析するなどのことが可能になるからである [2]．ノードの類似度について今までさまざまな手法が提案されているが，*Personalized PageRank (PPR)* は最もよく利用されているノードの類似度である [3]．PPR による類似度はグラフを問い合わせノードを起点にランダムウォークしたのちの定常状態における確率に対応する．PPR は繰り返し計算により求めることができるが，その計算においてスケーリングパラメータと呼ばれる確率 c は問い合わせノードにランダムウォークが戻る確率である [4]．PPR に対する検索は任意の与えられたグラフに対してアドホックに行えることが望ましい．アドホックな検索とは，検索を行うごとに検索対象のグラフやパラメータを変えながら検索を行うことである．それには二つの理由がある．まず初めの理由として多くのアプリケーションにおいてグラフは動的に変化する性質があるため，問い合わせノードが与えられる時点までグラフの構造が不明であることがあげられる．また二つ目の理由としてスケーリングパラメータ c が PPR の類似度のスコアに影響するためアプリケーションによって適切なスケーリングパラメータが異なることがあげられる [5]．しかしノード数を N，エッジ数を M，繰り返し計算回数を T としたとき PPR の計算コストは $O((N+M)T)$

† 日本電信電話株式会社　NTT ソフトウェアイノベーションセンタ，武蔵野市
NTT Software Innovation Center, NTT Corporation, Musashino-shi, 180–8585 Japan
†† 日本電信電話株式会社　NTT サービスエボリューション研究所，横須賀市
NTT Service Evolution Laboratories, NTT Corporation, Yokosuka-shi, 239–0847 Japan
††† 大阪大学，吹田市
Osaka University, Suita-shi, 565–0871 Japan
a) E-mail: fujiwara.yasuhiro@lab.ntt.co.jp
DOI:10.14923/transinfj.2014DEP0004

であり，グラフが大規模になるとアドホックに検索を行う計算時間が大きく増大してしまうという問題がある．

PPR の計算コストを低減するために今までさまざまな手法が提案されてきた [6]〜[12]．しかしいずれの手法も事前計算が必要であったりまたは検索結果が近似であったりなどの問題から多くのアプリケーションにおいてアドホックに検索を行うには適していない．そのため本論文では任意の与えられたグラフに対して PPR に対してアドホックに top-k 検索を正確かつ高速に行う問題に取り組む．実際の多くのアプリケーションにおいて PPR のスコアを正確に計算することより PPR に基づき正確にランキングを行うことのほうが重要である [4]．そのため本論文では正確にランキングを計算することを目的とする．

本論文は国際会議で過去に速報的に発表した論文 [13] と同じ内容であるが，本論文はこの論文をジャーナルにした正式版である．

本論文では正確なランキングに基づく top-k 検索を行う手法 *Castanet* を提案する．提案手法は検索コストを低減するために (1) 繰り返し計算の中で PPR のスコアの下限値と上限値を計算し，(2) 繰り返し処理の中で top-k 検索を行うのに不必要なノードを枝狩りする．本論文では提案手法を既存手法と比較し，提案手法の有効性を確認した．本論文の構成は以下のとおりである．**2.** で関連研究を述べる．**3.** で本論文における背景知識を述べる．**4.** で提案手法について述べる．**5.** で実験結果について述べる．**6.** で本論文をまとめる．

2. 関 連 研 究

PPR についての既存研究は行列分解に基づく手法とモンテカルロ法に基づく手法の二つにわけることができる．

Tong らは行列分解に基づく手法として B_LIN と NB_LIN を提案した [12]．これらの手法は固有値分解によって近似的に PPR のスコアを計算するものである．Fujiwara らは LU 分解及び QR 分解を用いて高速に PPR に対して検索を行う手法を提案した [10], [11]．これらの手法は Tong らの手法と異なり正確に PPR に対して検索を行えるというメリットがある．しかしこれらの行列分解に基づく手法は検索を行う前に行列分解を行う必要があるので，アドホックに検索を行うことには適していない．

Fogaras らはモンテカルロ法を用いて PPR を高速に計算する手法を提案した [9]．彼らの手法は fingerprint と呼ばれる問い合わせノードを起点とするランダムウォークを，近似の PPR のスコアを計算する事前に求める．検索時には事前に計算した fingerprint における終点の分布を用いて PPR のスコアを近似する．Bahmani も同様にランダムウォークを事前に計算する手法を提案した [8]．Avrachenkov らは PPR に対する検索を高速に行う手法として MC Complete Path を提案した．この手法は検索時にアドホックにランダムウォークを計算し高速に PPR の近似のスコアを求める．しかしモンテカルロ法による手法における精度はランダムウォークの回数に依存し，高速性と検索の精度のトレードオフがあるという問題点がある．

我々は過去に PageRank に対する高速な検索手法として F-Rank を提案した [14]．提案手法は F-Rank と異なり，ノードのランキングを正確に行いつつ top-k 検索を行う事が挙げられる．すなわち F-Rank は top-k 検索を行うが，検索結果においてノードのランキングは行わない．また提案手法は F-Rank と異なり，問い合わせノードからのホップ数に基づいた部分グラフを繰返し計算の中で計算し，計算対象となるノードを削減している．Personalized PageRank は PageRank と異なり，問い合わせノードに基づいてノードのランキングを行うが，提案手法は Personalized PageRank のこの性質を用いて，高速な検索を可能にしている．

3. 前 準 備

まずは本論文で用いる記号を定義し，必要となる背景知識を説明する．表 1 に主な記号とその定義を示す．PPR はグーグルで用いられている PageRank と同様に「ランダムサーファモデル」に基づき類似度を計算する．グーグルの PageRank はウェブグラフ全体におけるノードの重要度を計算するためにグラフのリンク構造を用いるが，Jeh らによって提案された PPR はこのグラフ全体におけるノードの重要度をユーザごとに個別化するという考えに基づいている [3]．すなわちユーザごとに複数のノードを任意に設定し，ノードごとに任意に設定された優先度に基づいてノードの重要度を計算する．なおノードごとの優先度は和が 1 になるように正規化される．ここで設定されたノードを問い合わせノードとすると各ノードの重要度は問い合わせノードに対する類似度と捉えることができる．端的に PPR における類似度はランダムウォークの定常状

表 1　主な記号とその定義
Table 1　Definitions of main symbols.

記号	定義
G	問い合わせ対象のグラフ
N	グラフにおけるノード数
M	グラフにおけるエッジ数
T	オリジナルの手法における繰り返し計算回数
c	スケーリングパラメータ，$0 < c < 1$
k	解ノードの数
V	G におけるノードの集合
E	G におけるエッジの集合
Q	問い合わせノードの集合
S	選択ノードの集合
R	選択ノードに到達可能なノードの集合
L	候補ノードの集合
D	ランキングが決定したノードの集合
A	解ノードの集合
$C[u]$	ノード u に隣接するノードの集合
$\mathbf{q}$	$N \times 1$ である問い合わせノードベクトル
$\mathbf{s}$	PPR による $N \times 1$ の類似度ベクトル
$\mathbf{W}$	$N \times N$ の列が正規化されたグラフの隣接行列

態における確率に対応している．PPR の各ステップでは現在のノードからその隣接ノードの一つを確率 c で選択し，また確率 $1-c$ で問い合わせノードの一つに優先度に従いジャンプする．V と E をそれぞれグラフのノードの集合とエッジの集合とすると問い合わせの対象のグラフは $G = \{V, E\}$ となる．また $\mathbf{s}$ を $N \times 1$ である PPR のスコアベクトルとすると，ノード u のスコアはベクトルの u 番目の要素 $s[u]$ と表現される．$\mathbf{W}$ は列の合計が 1 に正規化されたグラフの隣接行列であり，その要素 $W[u, v]$ はノード v からノード u にランダムウォークが遷移する確率となる．すなわち隣接行列 $\mathbf{W}$ において行がエッジの終点に対応し，列がエッジの始点に対応する．PPR のスコアは以下の式を再帰的に計算することによって得ることができる．

$$\mathbf{s} = c\mathbf{W}\mathbf{s} + (1-c)\mathbf{q} \tag{1}$$

なおここで $\mathbf{q}$ は問い合わせノードのベクトルであり，その u 番目の値 $q[u]$ は問い合わせノードとしての優先度に対応する．つまりノード u がもし問い合わせノードでなければ $q[u] = 0$ となる．問い合わせノードの優先度はその合計が 1 になるように正規化されている．

しかしこの再帰的に計算を行うオリジナルの手法は上位 k 個の検索を行うには適切ではない．それはこの手法では各繰り返し計算において全てのノードのスコアを更新する必要があるからである．更に上位 k 個を正確なランキングで検索するためにオリジナルの手法ではスコアが収束するまで繰り返し計算を行い正確なスコアを計算する必要があるが，実際に使われている多くのアプリケーションにおいてはランキングが重要であり必ずしも正確なスコアは必要ではない．オリジナルの手法は $O((N+M)T)$ の計算コストが必要であり，大規模なグラフに対して高速に検索を行うことが難しい．そのため高速に検索を行う手法が必要となる．

4. 提案手法

まずこの章ではまず提案手法の概要を説明し，それからその詳細を述べる．

4.1 手法概要

3. で説明したようにオリジナルの手法は定常状態における確率を計算する必要がある．この手法はグラフ全体を繰り返し用いてスコアを計算するため高い計算コストが必要となる．上位 k 個のノードを高速に求めるため，提案手法は全てのノードを再帰的に計算するのではなく選択されたノードに対してのみ再帰的に類似度の推定値を更新する．すなわち上位 k 個のノードの検索において提案手法は全てのノードを繰り返し計算することを避けることができ，高速な処理が可能になる．

類似度の推定値を高速に更新するために提案手法はグラフ全体から不要なノードとエッジを枝刈りし，動的に部分グラフを構築する．推定値を計算するために必要なランダムウォークの確率は部分グラフから計算できるため，提案手法は高速に推定値を更新することができる．不要なノードとエッジを特定するために提案手法は類似度の下限値と上限値を推定する．繰り返し計算の中で得られる部分グラフはグラフ全体より小さいため，高速に解ノードを検索することができる．

この部分グラフを用いる手法はオリジナルの手法と比較して繰り返し計算回数を減らせるという利点がある．オリジナルの手法は正確な類似度を計算するために類似度が収束するまで繰り返し計算を行う必要がある．一方提案手法はノードのランキングを下限値と上限値から決定することができ，ランキングが決定されたノードに対して繰り返し計算を行わない．そのため提案手法は全てのノードのランキングが下限値と上限値から決定すれば繰り返し計算を打ち切る．結果的に提案手法は類似度の収束を待つことなく計算を打ち切り，その結果，オリジナルの手法より繰り返し計算回数を少なくすることができる．

更に推定値を求める手法は部分グラフの構造と繰り返し計算回数を自動的に決定できるという利点がある．提案手法は必要となる内部パラメータはないため，ユーザは容易に PPR に対する検索を行うことができる．

4.2 類似度の推定

次に類似度の推定値の計算方法について述べる．まず推定値の計算に必要となる記号について述べ，それから具体的な計算方法とその理論的性質を述べる．

4.2.1 記号の定義

推定値を求めるために長さが i で問い合わせノードの一つから始まりノード u に到達するランダムウォークの確率 $p_i[u]$ を用いる．このランダムウォークにおいては一定確率 $1-c$ で問い合わせノードにジャンプすることはない．ここで $N \times 1$ のベクトル $\mathbf{p}_i$ をその u 番目の要素が $p_i[u]$ に対応するベクトルとする．$\mathbf{p}_i$ は隣接行列の i 乗から $\mathbf{p}_i = \mathbf{W}^i\mathbf{q}$ と計算することができる．ここで明らかに $\mathbf{p}_i = \mathbf{W}\mathbf{p}_{i-1}$ であるため，長さ i のランダムウォークの確率 $\mathbf{p}_i$ は $\mathbf{p}_{i-1}$ から以下のように逐次的に計算できる．

$$p_i[u] = \begin{cases} q[u] & (i=0) \\ \sum_{v \in C[u]} W[u,v] p_{i-1}[v] & (i \neq 0) \end{cases} \quad (2)$$

ここで $C[u]$ はノード u に入るエッジの始点の集合とする．すなわち $C[u]$ は元のグラフにおいてノード u に対して直接隣接するノードのうちエッジの終点がノード u になるノードの集合である．$i(=0,1,2,\ldots)$ 番目の繰り返し計算において提案手法は選択ノードの集合 S_i を計算し，そのノード集合に対して類似度の下限値と上限値を推定する．選択ノードの集合はグラフ全体のノード集合に初期化される．すなわち $S_0 = V$ である．また S_i は常に S_{i-1} に含まれるように設定される．結果的に $V = S_0 \supseteq S_1 \supseteq \ldots \supseteq S_i$ となる．具体的なノードの選択方法は後に述べる．類似度の上限値を計算するためにノード集合 R_i を用いる．R_i はノード集合 S_i のいずれかのノードへ到達可能なノードの集合である．ここでノード u がノード v へ到達可能とはオリジナルのグラフにおいてノード u から v へのパスがあるということである．ここでノード u からノード u へは到達可能であるため，明らかに $S_i \subseteq R_i$ であり，またもし $u \in S_i$ であれば $C[u] \subseteq R_i$ である．また $p_i[R_i]$ は長さが i のランダムウォークが問い合わせノードで始まり R_i のいずれかのノードに到達する確率で $p_i[R_i] = \sum_{u \in R_i} p_i[u]$ と計算される．上限値を計算するために $W_{max}[u]$ を用いるが，これはノード u に入るエッジの最大の重みである．すなわち $W_{max}[u] = \max\{W[u,v] : v \in V\}$ である．

4.2.2 類似度の定義

類似度の下限値は各繰り返し計算において以下のようにランダムウォークの確率を用いて計算する．

[**定義** 1]（下限値） i 番目の繰り返し計算においてノード u の下限値 $\underline{s}_i[u]$ は以下のように計算する．

$$\underline{s}_i[u] = \begin{cases} (1-c)p_i[u] & (i=0) \\ \underline{s}_{i-1}[u] + (1-c)c^i p_i[u] & (i \neq 0) \end{cases} \quad (3)$$

式 (3) が下限値をもつことは次の段落で示す．この定義は（1）もし $i=0$ であれば下限値はランダムウォークの確率とスケーリングパラメータから計算でき，（2）もしそうでなければ下限値 $\underline{s}_i[u]$ をランダムウォークの確率とスケーリングパラメータから逐次的に計算できることを示している．またこの定義はもしランダムウォークの確率 $p_i[u]$ が計算されていれば，下限値は $O(1)$ で計算できることも示している．

この下限値について以下の補助定理を示す．

[**補助定理** 1]（下限値） 集合 S_i に含まれるいずれのノードに対して $\underline{s}_i[u] \leq s[u]$ が成り立つ．

証明 補助定理 1 を証明するためにまず $\lim_{i\to\infty}(c\mathbf{W})^i = \mathbf{0}$ が成り立つことを示す．すなわちまず $(c\mathbf{W})^i$ が収束後において零行列になることを示す．$0 < c < 1$ であるため明らかに $\lim_{i\to\infty} c^i = 0$ である．ここで行列 $\mathbf{W}$ は列が正規化されたグラフの隣接行列であるため，その i 乗である $\mathbf{W}^i$ は長さが i のランダムウォークの確率に対応する．すなわち $\mathbf{W}^i$ のいずれの要素も 1 より大きくなることはなく 0 より小さくなることはない．そのため

$$\lim_{i\to\infty}(c\mathbf{W})^i = \lim_{i\to\infty} c^i \lim_{i\to\infty} \mathbf{W}^i = \mathbf{0} \quad (4)$$

となる．次に補助定理 1 を示す．式 (1) から PPR の類似度は以下のように計算できる．

$$\mathbf{s} = (1-c)(\mathbf{I} - c\mathbf{W})^{-1}\mathbf{q} \quad (5)$$

ここで $\mathbf{I}$ は単位行列であり，また $(\mathbf{I} - c\mathbf{W})^{-1}$ は $\mathbf{I} - c\mathbf{W}$ の逆行列である．$\lim_{i\to\infty}(c\mathbf{W})^i = \mathbf{0}$ であるため，$(c\mathbf{W})^0 = \mathbf{I}$ とすれば以下の式が成り立つ [15]．

$$(\mathbf{I} - c\mathbf{W})^{-1} = \sum_{j=0}^{\infty}(c\mathbf{W})^j \quad (6)$$

式 (5) と (6) から以下の式が成り立つ．

$$\mathbf{s}=(1-c)\left\{\sum_{j=0}^{\infty}(c\mathbf{W})^j\right\}\mathbf{q}=(1-c)\sum_{j=0}^{\infty}c^j\mathbf{p}_j \quad (7)$$

式 (7) は $\mathbf{s}$ の u 番目の要素 $s[u]$ が以下のように計算できることを示している．

$$s[u]=(1-c)\sum_{j=0}^{\infty}c^j p_j[u] \quad (8)$$

式 (3) から $\underline{s}_i[u]$ は以下のように計算できる．

$$\begin{aligned}\underline{s}_i[u]&=\underline{s}_{i-1}[u]+(1-c)c^i p_i[u]\\&=\underline{s}_{i-2}[u]+(1-c)c^i p_i[u]+(1-c)c^{i-1}p_{i-1}[u]\\&=(1-c)c^i p_i[u]+\ldots+(1-c)c^0 p_0[u]\end{aligned} \quad (9)$$

そのため $c>0$，$1-c>0$，$p_j[u]\geq 0$ であるため，

$$\underline{s}_i[u]=(1-c)\sum_{j=0}^{i}c^j p_j[u]\leq(1-c)\sum_{j=0}^{\infty}c^j p_j[u]=s[u] \quad (10)$$

となる．よって成り立つ． □

類似度の上限値を計算するために定義 1 による下限値を利用する．上限値の定義を以下に示す．

[**定義 2**]（上限値） i 番目の繰り返し計算においてノード u の上限値 $\overline{s}_i[u]$ は以下のように定義される．

$$\overline{s}_i[u]=\underline{s}_i[u]+c^{i+1}W_{max}[u]\,p_i[R_i] \quad (11)$$

この定義はノード u の上限値が c と $W_{max}[u]$ と $p_i[R_i]$ を用いることにより $O(1)$ で計算できることを示している．この上限値の性質を述べるために以下の補助定理を示す．

[**補助定理 2**]（上限値） 集合 S_i に含まれるいずれのノードに対して $\overline{s}_i[u]\geq s[u]$ が成り立つ．

証明 この補助定理を示すためにまず数学的帰納法を用いて $p_{i+j}[R_{i+j}]\leq p_i[R_i]$ $(j=0,1,\ldots)$ が全ての繰り返し計算において成り立つことを示す．

もし $j=0$ であれば明らかに $p_{i+j}[R_{i+j}]=p_i[R_i]$ である．また $j\neq 0$ すなわち $j\geq 1$ であるとき，$p_{i+j-1}[R_{i+j-1}]\leq p_i[R_i]$ が成り立つと仮定する．ノードは $S_{i+j}\subseteq S_{i+j-1}$ が成り立つように選択されているため，明らかに $R_{i+j}\subseteq R_{i+j-1}$ となる．そのため式 (2) から

$$\begin{aligned}p_{i+j}[R_{i+j}]&=\sum_{v\in R_{i+j}}p_{i+j}[v]\\&=\sum_{v\in R_{i+j}}\sum_{w\in C[v]}W[v,w]p_{i+j-1}[w]\\&\leq\sum_{w\in R_{i+j-1}}\sum_{v\in R_{i+j-1}}W[v,w]p_{i+j-1}[w]\\&\leq\sum_{w\in R_{i+j-1}}p_{i+j-1}[w]=p_{i+j-1}[R_{i+j-1}]\end{aligned} \quad (12)$$

となる．これは $v\in R_{i+j}$ であるようなノード v に対して $C[v]\subseteq R_{i+j}\subseteq R_{i+j-1}$ であり，$\mathbf{W}$ は列が正規化されたグラフの隣接行列だからである．なお $C[v]\subseteq R_{i+j-1}$ は $v\in R_{i+j}$ かつ $v\notin S_{i+j}$ であるようなノード v に成り立つ．これは $C[v]\subseteq C[v]+S_{i+j}\subseteq R_{i+j}\subseteq R_{i+j-1}$ だからである．そのため $p_{i+j}[R_{i+j}]\leq p_{i+j-1}[R_{i+j-1}]\leq p_i[R_i]$ となる．よって数学的帰納法より成り立つ．

次に補助定理 2 が成り立つことを示す．式 (2) と (8) と (10) からノード u の PPR の類似度 $s[u]$ は以下のように計算できる．

$$\begin{aligned}s[u]&=(1-c)\sum_{j=0}^{i}c^j p_j[u]+(1-c)\sum_{j=i+1}^{\infty}c^j p_j[u]\\&=\underline{s}_i[u]+(1-c)\sum_{j=i+1}^{\infty}\sum_{v\in C[u]}c^j W[u,v]p_{j-1}[v]\end{aligned} \quad (13)$$

ここで $W[u,v]\leq W_{max}[u]$ であり，$u\in S_i$ であるようなノード u に対して $C[u]\subseteq R_{j-1}$ であるため，

$$\begin{aligned}s[u]&\leq\underline{s}_i[u]+(1-c)W_{max}[u]\sum_{j=i+1}^{\infty}c^j\sum_{v\in R_{j-1}}p_{j-1}[v]\\&=\underline{s}_i[u]+(1-c)W_{max}[u]\sum_{j=i+1}^{\infty}c^j p_{j-1}[R_{j-1}]\end{aligned} \quad (14)$$

となる．$\sum_{j=i+1}^{\infty}c^j p_{j-1}[R_{j-1}]=\sum_{j=0}^{\infty}c^{i+j+1}p_{i+j}[R_{i+j}]$ であり，また上記のとおり $p_{i+j}[R_{i+j}]\leq p_i[R_i]$ であるため，

$$\begin{aligned}s[u]&\leq\underline{s}_i[u]+(1-c)W_{max}[u]\,p_i[R_i]\sum_{j=0}^{\infty}c^{i+j+1}\\&=\underline{s}_i[u]+(1-c)W_{max}[u]\,p_i[R_i]\frac{c^{i+1}-c^{\infty}}{1-c}\end{aligned}$$

$$\leq \underline{s}_i[u] + c^{i+1} W_{max}[u]\, p_i[R_i] = \overline{s}_i[u] \quad (15)$$

となる．よって成り立つ． □

定義 1 と 2 に示すとおり，下限値と上限値は各繰り返し計算においてランダムウォークの確率から推定される．これらの推定値は繰り返し計算が進むごとに推定の精度が上がっていくという性質がある．この性質を示すために以下の補助定理を示す．

[**補助定理 3**]（精度向上） i 番目の繰り返し計算において $\underline{s}_i[u] \geq \underline{s}_{i-1}[u]$ であり $\overline{s}_i[u] \leq \overline{s}_{i-1}[u]$ となる．

証明 まず $\underline{s}_i[u] \geq \underline{s}_{i-1}[u]$ であることを示す．式 (3) から明らかに $\underline{s}_i[u] - \underline{s}_{i-1}[u] = (1-c)c^i p_i[u] \geq 0$ となる．次に $\overline{s}_i[u] \leq \overline{s}_{i-1}[u]$ であることを示す．式 (3) と (11) から $\overline{s}_i[u] - \overline{s}_{i-1}[u]$ は以下のように計算できる．

$$\begin{aligned}
&\overline{s}_i[u] - \overline{s}_{i-1}[u] \\
&= \underline{s}_i[u] - \underline{s}_{i-1}[u] + c^i W_{max}[u](c p_i[R_i] - p_{i-1}[R_{i-1}]) \\
&= c^i\{(1-c)p_i[u] + W_{max}[u](c p_i[R_i] - p_{i-1}[R_{i-1}])\}
\end{aligned} \quad (16)$$

式 (2) から

$$\begin{aligned}
p_i[u] &= \sum_{v \in C[u]} W[u,v]\, p_{i-1}[v] \\
&\leq W_{max}[u] \sum_{v \in R_{i-1}} p_{i-1}[v] \\
&= W_{max}[u]\, p_{i-1}[R_{i-1}]
\end{aligned} \quad (17)$$

となる．式 (12) から $p_i[R_i] \leq p_{i-1}[R_{i-1}]$ であるため，以下のようになる．

$$\begin{aligned}
&\overline{s}_i[u] - \overline{s}_{i-1}[u] \\
&\leq c^i W_{max}[u]\{(1-c)p_{i-1}[R_{i-1}] + c p_{i-1}[R_{i-1}] - p_{i-1}[R_{i-1}]\} \\
&= 0
\end{aligned} \quad (18)$$

よって成り立つ． □

後に説明するように，補助定理 3 により提案手法は収束するまで類似度を計算することなく正確なランキングで上位 k 個のノードを検索することができる．下限値と上限値は以下のように正確な類似度に収束するという性質がある．

[**補助定理 4**]（推定値の収束値） 収束後において下限値と上限値は正確な類似度と等しくなる．すなわち $\underline{s}_\infty[u] = \overline{s}_\infty[u] = s[u]$ である．

証明 まず $\underline{s}_\infty[u] = s[u]$ であることを示す．式 (10) から $\underline{s}_\infty[u] = (1-c)\sum_{j=0}^{\infty} c^j p_j[u] = s[u]$ であることは明らかである．次に $\overline{s}_\infty[u] = s[u]$ であることを示す．式 (11) から $\overline{s}_\infty[u] = \underline{s}_\infty[u] + c^\infty W_{max}[u]\, p_\infty[R_\infty]$ である．$c^\infty = 0$ であり $0 \leq W_{max}[u] \leq 1$ であり $0 \leq p_\infty[R_\infty] \leq 1$ であるため，$c^\infty W_{max}[u]\, p_\infty[R_\infty] = 0$ となる．そのため $\overline{s}_\infty[u] = \underline{s}_\infty[u] = s[u]$ となる．よって成り立つ． □

後に説明するように，補助定理 4 により提案手法は正確なランキングを行える．

4.3 部分グラフによる検索

提案手法では部分グラフを繰り返し計算の中で動的に構築しランダムウォークの確率を高速に計算する．選択ノードに対する下限値と上限値はランダムウォークの確率を用いて計算する．ここではまずノードの選択方法について述べ，そして部分グラフの定義を示す．

4.3.1 選択ノード

提案手法ではあるノードがもし（1）解ノードになる可能性があり，かつ（2）推定値を用いても解ノードとしてのランキングが決定できないのであれば，そのノードを選択する．θ_i を i 番目の繰り返し計算における k 番目に高い下限値とすると，i 番目の繰り返し計算における解の候補ノード集合 L_i は以下のように定義される．

[**定義 3**]（候補ノード） i 番目の繰り返し計算において正確な類似度が θ_i より大きくなりうる候補ノードの集合 L_i は以下のように定義される．

$$L_i = \{u \in V : \overline{s}_i[u] \geq \theta_i\} \quad (19)$$

この定義はもしあるノードの上限値が θ_i 以上であればそのノードは候補ノードとなることを示している．これはもしあるノードの上限値がもし θ_i 未満であればそのノードの正確な類似度は θ_i 以上になることはなく，解ノードになり得ないからである．後に示すように候補ノードの集合は繰り返し計算の中で単調減少する性質がある．i 番目の繰り返し計算において解ノードとしてのランキングが決定したノードの集合 D_i は以下のように定義される．

[**定義 4**]（決定ノード） もし $|L_i| = k$，すなわち候補ノードの数が k であれば，i 番目の繰り返し計算において解ノードとしての正確なランキングが決定しているノードの集合 D_i は以下のように定義される．

$$\begin{aligned}
D_i = \{&u \in L_i : \underline{s}_i[u] > \overline{s}_i[v] \\
&\text{or } \overline{s}_i[u] < \underline{s}_i[v], u \neq v, \forall v \in L_i\}
\end{aligned} \quad (20)$$

もし $|L_i| \neq k$ であれば D_i は以下のように定義される．

$$D_i = \emptyset \tag{21}$$

定義 4 はあるノード u を，もし（1）解ノードの数が k でありかつ（2）下限値または上限値が $\underline{s}_i[u]$ と $\overline{s}_i[u]$ の間にあるノード $v(\neq u)$ が存在しなければ決定ノードとする．すなわち下限値または上限値が $\underline{s}_i[u]$ と $\overline{s}_i[u]$ の間にあるノードが存在すれば，ノード u は決定ノードとならない．定義 4 においてもし $|L_i| = k$ であれば決定ノード D_i を計算するコストは $O(k \log k)$ である．これは D_i が L_i におけるノードを並び変えることによって得られるからである．また $|L_i| \neq k$ であれば式 (21) から D_i を計算する必要はない．ノード集合 L_i と D_i の定義から選択ノードの集合 S_i は以下のように定義される．

[**定義** 5]（選択ノード） i 番目の繰り返し計算において下限値と上限値を計算する選択ノード S_i は以下のように定義される．

$$S_i = \begin{cases} V & (i = 0) \\ L_{i-1} \backslash D_{i-1} & (i \neq 0) \end{cases} \tag{22}$$

この式において $L_{i-1}\backslash D_{i-1}$ は集合 L_{i-1} から D_{i-1} を引いた差集合で $\{u \in L_{i-1} : u \notin D_{i-1}\}$ と計算される．

式 (5) から提案手法はまず全てのノードに対して推定値を計算し，あるノードがもし（1）候補ノードでないか，または，(2) ランキングが決定していればそのノードの推定値を更新しない．すなわち提案手法は推定値を候補ノードでかつランキングが決定していない場合のみ更新する．

選択ノードの性質を示すために集合 L_i と D_i の幾つかの補助定理を示す．まず解ノードを A としたとき，候補ノード L_i の性質を以下のとおりである．

[**補助定理** 5]（L_i の単調減少） 繰り返し計算の中において候補ノード L_i は単調減少する．すなわち $L_i \subseteq L_{i-1}$ である．

証明 θ_i を i 番目の繰り返し計算における k 番目に高い下限値とすると，補助定理 3 に示すとおり下限値は単調増加する性質があるため $\theta_i \geq \theta_{i-1}$ となる．そのため（1）もしノード u が L_i に含まれれば $\overline{s}_{i-1}[u] \geq \overline{s}_i[u] \geq \theta_i \geq \theta_{i-1}$ であるためノード u は集合 L_{i-1} に含まれ，(2) そうでなければ $\theta_i > \overline{s}_{i-1}[u] \geq \overline{s}_i[u] \geq \theta_{i-1}$ となりうるためノード u は集合 L_{i-1} に含まれる可能性がある． □

[**補助定理** 6]（L_i の収束） 収束後において候補ノードは解ノードと等しくなる．すなわち $L_\infty = A$ である．

証明 θ を解ノードにおける k 番目に高い正確な類似度とすると，補助定理 4 から収束後において $\theta_\infty = \theta$ となる．そのため補助定理 4 から

$$L_\infty = \{u \in V : \overline{s}_\infty[u] \geq \theta_\infty\} = \{u \in V : s[u] \geq \theta\} = A \tag{23}$$

となる． □

補助定理 3 から集合 L_i を計算するには全てのノードが必要となるが，繰り返し計算においてより効率的に集合 L_i を計算できる．もし $i \neq 0$ で $|L_{i-1}| \neq k$ であれば，集合 L_i は以下のように計算できる．

$$L_i = \{u \in L_{i-1} : \overline{s}_i[u] \geq \theta_i\} \tag{24}$$

式 (24) は式 (19) において V を L_{i-1} に入れ替えることによって得られる．すなわち集合 L_{i-1} から集合 L_i を逐次的に計算できる．これはもしあるノードが集合 L_{i-1} に含まれてなければ補助定理 5 よりそのノードが集合 L_i に含まれることはないからである．更にもし $i \neq 0$ で $|L_{i-1}| = k$ であれば集合 L_i は以下のように計算できる．

$$L_i = L_{i-1} \tag{25}$$

これは補助定理 5 と 6 から $L_i \subseteq L_{i-1}$ であり集合 L_i は集合 A に収束するからである．

補助定理 5 と 6 から決定ノードに対する以下の性質を示す．

[**補助定理** 7]（D_i の単調増加） 決定ノードの集合 D_i は単調増加する性質がある．すなわち i 番目の繰り返し計算において $D_i \supseteq D_{i-1}$ が成り立つ．

証明 まず $|L_{i-1}| \neq k$ の場合成り立つことを示す．定義 4 からこの場合 $D_{i-1} = \emptyset \subseteq D_i$ である．次に $|L_{i-1}| = k$ の場合に成り立つことを示す．この場合補助定理 5 と 6 から $L_i = L_{i-1} = A$ となる．もしノード u が集合 D_{i-1} に含まれれば，そのノードは必ず集合 D_i に含まれる．これは（1）補助定理 3 より推定値は繰り返し計算の中で精度が向上する性質があり，(2) $L_i = L_{i-1}$ だからである．もしノード u が集合 D_{i-1} に含まれていなければ，推定値は精度が向上する性質があるためそのノードは D_i に含まれる可能性

がある．□

[**補助定理** 8]（D_i の収束） 収束後において $D_\infty = A$ となる．すなわち決定ノードの集合は解ノードの集合と等しくなる．

証明 収束後において補助定理 4 から $\underline{s}_\infty[u] = \overline{s}_\infty[u] = s[u]$ であり，補助定理 6 から $L_\infty = A$ である．そのため定義 4 より

$$\begin{aligned} D_\infty &= \{u \in L_\infty : \underline{s}_\infty[u] > \overline{s}_\infty[v] \\ &\qquad \text{or } \overline{s}_\infty[u] < \underline{s}_\infty[v], u \neq v, \forall v \in L_\infty\} \\ &= \{u \in A : s[u] > s[v] \\ &\qquad \text{or } s[u] < s[v], u \neq v, \forall v \in A\} = A \end{aligned} \quad (26)$$

となる．そのため収束後において $D_\infty = A$ となる．□

補助定理 5 と 7 より集合 L_i と D_i はそれぞれ単調増加/減少する性質がある．そして補助定理 6 と 8 から集合 L_i と D_i は収束後において解ノードの集合 A と等しくなる．そのため選択ノードについて以下の性質が成り立つ．

[**補助定理** 9]（選択ノード） 選択ノードは単調減少する性質があり，また収束後において空集合となる．すなわち $S_i \subseteq S_{i-1}$ であり $S_\infty = \emptyset$ である．

証明 これは補助定理 5 と 6 と 7 と 8 から明らかである．□

定義 5 にあるとおり選択ノードはグラフの全ノードの集合に初期化されるため，選択ノードは $V = S_0 \supseteq S_1 \supseteq \ldots \supseteq S_i$ という性質がある．更に $S_\infty = \emptyset$ ということは更新するべきノードがないことを示しており，これは提案手法が有限の繰り返し回数で計算を打ち切ることを示している．

4.3.2 部分グラフの構築

ここでは推定値を高速に更新するために部分グラフを構築する方法を述べる．素直に推定値を更新するには各繰り返し計算においてグラフ全体を用いる方法がある．しかしこの方法は全てのノードに対してランダムウォークの確率を計算する必要になるので計算コストが高くなる問題がある．そのため提案手法は不必要なノードとエッジを枝刈りし，高速に推定値を更新する．まずここでは部分グラフの定義を述べてからその性質を述べる．また各繰り返し計算において逐次的に部分グラフを更新する方法も述べる．

定義 1 と 2 に見られるように，下限値と上限値は式 (2) で与えられるランダムウォークの確率を用いて計算できる．そのためランダムウォークの確率を高速に計算するために部分グラフを構築する．部分グラフを構築するために問い合わせノードからのホップ数が i 以内であるノードの集合 H_i を用いる（つまり H_i には問い合わせノードからのホップ数が i より大きくなるノードは含まれない.）．i 番目の繰り返し計算における部分グラフ G_i はノード集合 H_i を用いて以下のように計算される．

[**定義** 6]（部分グラフ） i 番目の繰り返し計算におけるグラフ G の部分グラフを $G_i = \{V_i, E_i\}$ とすると，V_i と E_i は以下のように定義される．

$$V_i = H_i \cap R_i \quad (27)$$

$$E_i = \{(u, v) \in E : u \in V_i, v \in V_i\} \quad (28)$$

ここで (u, v) はノード u から v のエッジである．

この定義は（1）もしオリジナルのグラフ G_i においてあるノードが問い合わせノードから i ホップ以内でありかつそのノードから問い合わせノードへ到達可能であれば，部分グラフはそのノードをもち，(2) もし部分グラフにおいて二つのノードがエッジでつながっていてかつそれらのノードがともに部分グラフにあれば，そのエッジは部分グラフにあることを示している．

またこの定義は問い合わせノードの数が少ないほど部分グラフの大きさが小さくなり，より高速に検索を行えることを示している．これは問い合わせノードの数が少ないほど，式 (27) で与えられる部分グラフにおけるノードの集合が小さくなるからである．問い合わせノードの数と検索時間の関係については **5.** で示す．

部分グラフの性質を示すために以下の補助定理を示す．

[**補助定理** 10]（部分グラフによる下限値と上限値） V_i と E_i から選択ノードの i 番目の繰り返し計算における下限値と上限値を計算できる．

証明 先に述べたとおり $p_i[u]$ は問い合わせノードから始まりノード u に到達する長さ i のランダムウォークの確率である．そのためもしあるノードが問い合わせノードから i ホップより離れていれば i 番目の繰り返し計算においてそのランダムウォークの確率は 0 になる．結果としてノードの下限値と上限値はノード集合 H_i とそれらのノード間のエッジ H_i から計算できる．また式 (2) と (3) と (11) からあるノードがもし選択ノードの集合 S_i に到達できないのであれば，そのノードのランダムウォークの確率は集合 S_i に含まれるノードの下限値と上限値に影響がない．そのためノード集合 R_i とそれらのノード間の

エッジ R_i から集合 S_i に含まれるノードの下限値と上限値は計算できる．結果的に i 番目の繰り返し計算において選択されたノードの下限値と上限値はノード集合 $V_i = H_i \cap R_i$ とそれらのノード間のエッジ $E_i = \{(u,v) \in E : u \in V_i, v \in V_i\}$ からのみから計算できる． □

この証明から選択ノードに対して部分グラフからランダムウォークの確率と推定値を計算できることがわかる．部分グラフを用いて以下のように効率的に下限値と上限値を計算する．

[**定義 7**]（部分グラフによる確率の計算） $C_i[u]$ を部分グラフ G_i におけるノード u に入るエッジの始点の集合としたときに，i 番目の繰り返し計算においてノード u のランダムウォークの確率 $p_i[u]$ を以下のように計算する．

$$p_i[u] = \begin{cases} q[u] & (i = 0) \\ \sum_{v \in C_i[u]} W[u,v] p_{i-1}[v] & (i \neq 0) \end{cases} \quad (29)$$

上記の式は式 (2) において $C[u]$ を $C_i[u]$ に置き換えることで得ることができる．上位 k 個のノードを高速に検索するために，上記の式を用いて部分グラフからランダムウォークの確率を計算する．ノードの下限値と上限値は定義 1 と 2 からそれぞれ計算する．素直に下限値と上限値を計算するにはグラフ全体を処理する必要があるが，部分グラフを用いることによって効率的に推定値を計算できる．

しかし定義 6 をそのまま適用するとグラフの全てのノードを用いて部分グラフを計算する必要があるため，効率的に部分グラフを構築できないという問題がある．そのためノードの集合 h_i とエッジの集合 e_i を用いて繰り返し計算において逐次的に部分グラフを構築する．ここで h_i は問い合わせノードからのホップ数が i であるノードの集合である．そのため $h_0 = H_0 = Q$ であり $H_i = \bigcup_{j=0}^{i} h_j = H_{i-1} + h_i$ である．また e_i はノードの集合 h_i と H_{i-1} の間のエッジの集合である．すなわち $H_i = H_{i-1} + h_i$ であるため $e_i = \{(u,v) \in E : u \in h_i, v \in H_{i-1}$ or $u \in H_{i-1}, v \in h_i$ or $u \in h_i, v \in h_i\}$ である．なおここで h_i と e_i はグラフ全体に対して問い合わせノードをルートノードとし 1 度幅優先探索を行えば得ることができる．そのため h_i と e_i は $O(N+M)$ の計算コストで得られる．h_i と e_i に対して以下の二つの補助定理を示す．

[**補助定理 11**]（部分グラフの包含） ノードの集合とエッジの集合をそれぞれ $V_i' = V_{i-1} + h_i$ と $E_i' = E_{i-1} + e_i$ とすると，もし $i \neq 0$ であれば $V_i \subseteq V_i'$ であり $E_i \subseteq E_i'$ となる．

証明 もし $i \neq 0$ であれば式 (27) より $H_i = H_{i-1} + h_i$ であり $R_i \subseteq R_{i-1}$ であるため，式 (27) より

$$V_i = (H_{i-1} + h_i) \cap R_i \subseteq (H_{i-1} + h_i) \cap R_{i-1} \quad (30)$$

となる．そのため

$$V_i \subseteq H_{i-1} \cap R_{i-1} + h_i \cap R_{i-1} \subseteq V_{i-1} + h_i = V_i' \quad (31)$$

となる．もし $i \neq 0$ であれば式 (28) より $V_i \subseteq V_{i-1} + h_i$ であるため，

$$E_i \subseteq \{(u,v) \in E : u \in (V_{i-1} + h_i), v \in (V_{i-1} + h_i)\} \quad (32)$$

となる．そのため $V_{i-1} = H_{i-1} \cap R_{i-1} \subseteq H_{i-1}$ であるため

$$\begin{aligned} E_i \subseteq & \{(u,v) \in E : u \in V_{i-1}, v \in V_{i-1}\} \\ & + \{(u,v) \in E : u \in H_{i-1}, v \in h_i \\ & \quad \text{or } u \in h_i, v \in H_{i-1} \text{ or } u \in h_i, v \in h_i\} \end{aligned} \quad (33)$$

となる．結果として $E_i \subseteq E_{i-1} + e_i = E_i'$ となる． □

補助定理 11 から繰り返し計算において逐次的に部分グラフを構築することができる．グラフ G_i' を $G_i' = \{V_i', E_i'\}$ とすると，$V_i' = V_{i-1} + h_i$ であり $E_i' = E_{i-1} + e_i$ であるため，グラフ G_i' は逐次的に構築することができる．すなわちグラフ G_i' はグラフ G_{i-1} に問い合わせノードから i ホップ離れたノードとそれにつながるエッジを足すことで得られる．ここで（1）上記の補助定理より $G_i \subseteq G_i'$ であり，(2) 集合 h_i と e_i には選択ノードの集合 S_i へのいかなるパスにも含まれないノードとエッジを含むため，幅優先探索によりグラフ G_i' における集合 S_i へのパスを幅優先探索で求めることで部分グラフ G_i を計算できる．

部分グラフを逐次的に構築するためのアルゴリズムを Algorithm 1 に示す．もし $i = 0$ であればアルゴリズムはノードの集合を $V_0 = Q$ に初期化し，エッジの集合を $E_0 = \{(u,v) \in E : u \in Q, v \in Q\}$ に初期化する（2〜3 行目）．これは式 (22) と (27) より $V_0 = H_0 \cap R_0 = Q \cap V = Q$ だからである．もしそうでなければ更新前のグラフ G_{i-1} からグラフ G_i' を

Algorithm 1 部分グラフ構築アルゴリズム

Input: G_{i-1}, 更新前の部分グラフ; h_i, ノードの集合; e_i, エッジの集合; S_i 選択ノード
Output: G_i, 更新後の部分グラフ

1: **if** $i = 0$ **then**
2: $V_0 := Q$;
3: $E_0 := \{(u, v) \in E : u \in Q, v \in Q\}$;
4: **else**
5: $V_i' := V_{i-1} + h_i$;
6: $E_i' := E_{i-1} + e_i$;
7: $G_i' := \{V_i', E_i'\}$;
8: G_i' において S_i へのパスを計算;
9: パスから V_i と E_i を計算;
10: **end if**
11: $G_i := \{V_i, E_i\}$;
12: **return** G_i;

逐次的に計算する（5〜7 行目）．そして集合 V_i と E_i をグラフ G_i' から幅優先探索を用いて計算する（8〜9 行目）．Algorithm 1 に見られるように部分グラフはオリジナルのグラフ全体を用いることなく計算できる．

4.4 検索アルゴリズム

部分グラフを用いた検索アルゴリズムを Algorithm 2 に示す．まず選択ノードの集合を初期化し（2 行目），部分グラフを計算する（5〜6 行目）．そして部分グラフに含まれるノードに対してランダムウォークの確率を計算する（7〜9 行目）．これはランダムウォークの確率が推定値の計算に必要だからである（補助定理 10）．そして選択ノードの集合に対して推定値を計算する（10〜13 行目）．もし $i = 0$ であれば定義 3 から候補ノードの集合 L_i を計算する（14〜15 行目）．そうでなければ集合 L_{i-1} を用いて L_i を逐次的に計算する（16〜22 行目）．式 (4) からもし $|L_i| \neq k$ であれば $D_i = \emptyset$ であるため，$|L_i| = k$ である場合のみ D_i を計算する（23〜24 行目）．そして選択ノードの集合を更新する（28 行目）．もし選択ノードの集合が空集合であれば繰返しを打ち切る（29 行目）．そしてノードのランキングを集合 D_i にあるノードを下限値または上限値で並び変えて計算する（30 行目）．最後に集合 D_i を解ノードとして出力する（31〜32 行目）．

Algorithm 2 にあるとおり，提案手法は検索における事前計算を必要としない．すなわち提案手法はアドホックに検索を行うことができる．また提案手法はユーザに内部パラメータの設定を求めることはない．そのためユーザは簡易に PPR に対する検索を行うことができる．

Algorithm 2 検索アルゴリズム

Input: G, グラフ; c, スケーリングパラメータ; k, 解ノードの数; Q, 問い合わせノード
Output: A, 解ノード

1: $i := -1$;
2: $S_0 := V$;
3: **repeat**
4: $i := i + 1$;
5: 幅優先探索により h_i と e_i を計算;
6: Algorithm 1 より G_i を計算;
7: **for** $u \in V_i$ となるノードに対して **do**
8: 式 (29) から G_i を用いて $p_i[u]$ を計算;
9: **end for**
10: **for** $u \in S_i$ となるノードに対して **do**
11: 式 (3) から $\underline{s}_i[u]$ を計算;
12: 式 (11) から $\overline{s}_i[u]$ を計算;
13: **end for**
14: **if** $i = 0$ **then**
15: 式 (19) から L_i を計算;
16: **else**
17: **if** $|L_{i-1}| \neq k$ **then**
18: 式 (24) から L_i を計算;
19: **else**
20: $L_i := L_{i-1}$;
21: **end if**
22: **end if**
23: **if** $|L_i| = k$ **then**
24: 式 (20) から D_i を計算;
25: **else**
26: $D_i := \emptyset$;
27: **end if**
28: $S_{i+1} := L_i \backslash D_i$;
29: **until** $S_{i+1} = \emptyset$
30: D_i に含まれるノードを並び変え;
31: $A := D_i$;
32: **return** A;

4.5 理論的解析

提案手法における検索結果と計算量について述べる．まず検索結果についての以下の定理を示す．

[**定理** 1]（検索の正確性） 提案手法は上位 k 個のノードを正確なランキングで計算する．

証明 Algorithm 2 は選択ノードの集合が空集合に収束するまで計算を行い集合 D_i を解ノードの集合として出力する．もし $S_{i+1} = \emptyset$ であれば定義 5 から $L_i = D_i$ である．補助定理 5 と 6 と 7 と 8 から $L_i \supseteq A$ であり $D_i \subseteq A$ であるため，$L_i = D_i = A$ if $L_i = D_i$ となる．そのためもし $S_{i+1} = \emptyset$ であれば $D_i = A$ となる．$D_i = A$ であるため収束後において集合 D_i におけるランキングは全て決定できる．そのため上位 k 個のノードを正確なランキングで計算できる． □

次に提案手法の計算量を述べる．まず l と t をそれぞれ選択ノードの平均数と提案手法における繰返し計

算回数とする．また n と m をそれぞれ部分グラフにおけるノードのエッジの平均個数とする．なおオリジナルの手法の計算コストは $O(N+M)T$ である．

[**定理 2**]（計算コスト） 提案手法が検索を行うのに必要な平均計算量は $O((l+n+m+k\log k)t+N+M)$ である．

証明 提案手法は，まず部分グラフを構築するが，このために必要な平均計算量は $O((n+m)t+N+M)$ である．これは (1) 計算量が $O((n+m)t)$ である幅優先探索を用いて各繰返し計算において部分グラフのノードとエッジを計算し，(2) h_i と e_i は $O(N+M)$ の平均計算量で得られるからである．またランダムウォークの確率と推定値は繰り返し計算において部分グラフを用いてそれぞれ $O((n+m)t)$ と $O(l\cdot t)$ の平均計算量で得られる．また候補ノードと決定ノードの集合 L_i と D_i はそれぞれ $O(l\cdot t)$ と $O(k\log k\cdot t)$ の計算量で得られる．収束後において解ノードとそのランキングは $O(k\log k)$ の計算量で得られる．結果として提案手法における平均計算量は $O((l+n+m+k\log k)t+N+M)$ となる． □

5. 評価実験

提案手法 Castanet について評価実験を行った．実験では Avrachenkov らのモンテカルロ法に基づく手法 [6] と，Fujiwara らの行列分解に基づく手法 [11] と，オリジナルの手法 [3] と比較を行った．この章において “Castanet”，“Monte”，“Matrix”，“Original” はそれぞれ上記の四つの手法による実験結果を示す．

実験では以下のデータを用いた．

- Notredame(注1): これはノートルダム大学におけるウェブのデータである．このグラフにおいてノードはウェブページでありエッジはそれらの間のハイパーリンクである．ノード数とエッジ数はそれぞれ 325,729 と 1,497,135 である．
- CNR(注2): CNR とはイタリアの研究組織の一つであり，このグラフは CNR におけるウェブグラフである．このグラフにおいてノード数は 325,557 であり，エッジ数は 3,216,152 である．
- Email(注3): このグラフはヨーロッパにある研究組織における電子メールのデータである．このグラフにおいてノードは電子メールのアドレスに対応し，ノード間にエッジがある場合はそれらに対応する電子メールアドレスの間で少なくとも 1 回以上電子メールのやり取りがあったことに対応する．このグラフにおいてノード数とエッジ数はそれぞれ 265,214 と 420,045 である．
- Social(注4): このグラフは Slashdot.org から得られたものである．ノードは Slashdot.org のユーザに対応し，エッジはユーザ間にやり取りがあることを表す．このグラフにおいてノード数は 82,144 であり，エッジ数は 549,202 である．

実験において問い合わせノードはランダムに三つ選択した．実験は CPU が 3.33 GHz の Intel Xeon サーバで行った．

5.1 検索時間

各手法の検索時間を調べた．図 1 に結果を示す．この図において提案手法の結果を “Castanet(k)” とし，ここで k を検索を行う解の個数とした．またスケーリングパラメータを $c=0.5$ としモンテカルロ法に基づく手法におけるランダムウォークの回数を 2,500,000 とした．既存手法において解の個数は検索時間に影響がなかった．これは (1) モンテカルロ法に基づく手法とオリジナルの手法では全てのノードのスコアを計算し，(2) 行列分解に基づく手法では事前の行列分解に必要な時間が支配的であったためである．表 2 に $c=0.5$ で $k=10$ としたときの提案手法とオリジナルの手法におけるパラメータの詳細を示す．また表 3 には Notredame をデータセットとし $c=0.5$ で $k=10$ としたときの各手法の検索時間の詳細を示す．

図 1 に示すとおり，提案手法はオリジナルの手法に

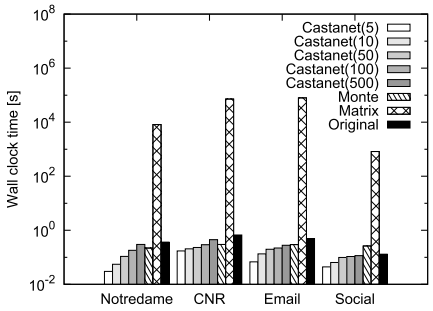

図 1 検索時間
Fig. 1 Search time.

(注1)：http://vlado.fmf.uni-lj.si/pub/networks/data/ND/NDnets.htm
(注2)：http://law.di.unimi.it/webdata/cnr-2000/
(注3)：http://snap.stanford.edu/data/email-EuAll.html
(注4)：http://snap.stanford.edu/data/soc-sign-Slashdot090221.html

表 2　各パラメータの値
Table 2　Score of each parameter.

パラメータ	データセット			
	Notredame	CNR	Email	Social
N	3.25×10^5	3.25×10^5	2.65×10^5	8.21×10^4
l	2.29×10^4	2.13×10^4	2.27×10^4	7.92×10^3
n	5.12×10^4	7.01×10^4	5.98×10^4	2.88×10^4
M	1.49×10^6	3.21×10^6	4.20×10^5	5.49×10^5
m	7.30×10^4	5.70×10^5	9.76×10^4	2.17×10^5
T	30.8	32.8	56.0	28.0
t	15.5	24.3	15.7	13.1

表 3　検索時間の詳細
Table 3　Breakdown of search time.

手法	検索時間 [ms]		
	事前処理	検索処理	合計
Castanet	–	5.52	5.52
Monte	–	21.8	21.8
Matrix	8.16×10^5	5.60×10^{-3}	8.16×10^5
Original	–	36.5	36.5

対して最大 90%ほど検索時間を削減している．またこの図から提案手法は解の個数 k が小さくなるほど検索時間が短くなることがわかる．これは解の個数 k が小さくなるほど定義 6 におけるノードの集合 H_i と R_i が小さくなるからである．もし $k = 500$ であればモンテカルロ法に基づく手法のほうが提案手法より高速にある場合があるが，実際のアプリケーションにおいて $k = 500$ になることはない [5], [16]．また表 3 に示すとおり行列分解に基づく手法の検索時間自体は提案手法より高速かも知れないが，事前計算時間も含めると全体としては提案手法のほうが高速である．

5.2　検索精度

モンテカルロ法に基づく手法に対する提案手法の特徴として検索の結果におけるノードのランキングが正確であることが挙げられる．モンテカルロ法に基づく手法はランダムウォークに回数と検索精度にあるため，この実験ではランダムウォークの回数を変化させ，モンテカルロ法に基づく手法と提案手法の比較を行った．Notredame をデータセットとしたときの検索時間と検索精度についてそれぞれ図 2 と図 3 にしめす．なおここで $c = 0.5$ で $k = 10$ とした．図 3 において検索精度の指標として average precision を用いた [17]．average precision は 0 から 1 の間の数値であり，もしオリジナルの検索結果と同じ結果となるのなら average precision は 1 になる．

図 2 に示すとおりランダムウォークの回数が増加するとモンテカルロ法に基づく手法の検索時間も増加する．更にランダムウォークの回数が適切に設定されていないと，モンテカルロ法に基づく手法はオリジナルの手法より検索時間が要する場合が出てくる．図 3 からわかるとおり提案手法はランキングが正確であることが保証されているので，その average precision は 1 であるが，モンテカルロ法に基づく手法の検索精度はそれより低い．またこの図よりモンテカルロ法に基づく手法はランダムウォークの回数を増やしても精度が向上しないことがわかる．これはこの実験条件においてモンテカルロ法に基づく手法は既に精度が収束しているためである．図 2 と図 3 から提案手法はモンテカルロ法に基づく手法より検索速度と検索精度において優れていることがわかる．

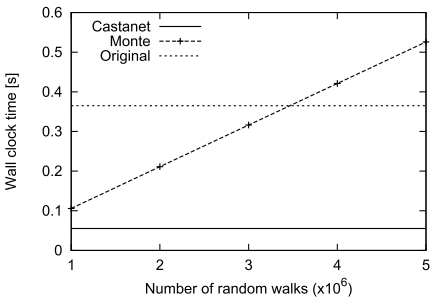

図 2　ランダムウォークの回数と検索時間
Fig. 2　Efficiency versus number of random walks.

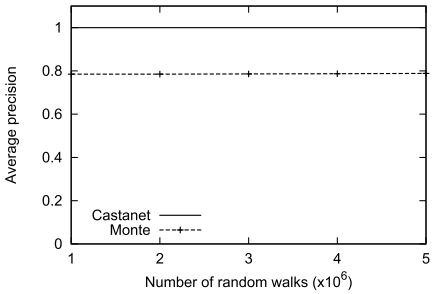

図 3　ランダムウォークの回数と検索精度
Fig. 3　Accuracy versus number of random walks.

5.3　複数のスケーリングパラメータに対する評価

1. で述べたとおり実際のアプリケーションにおいてはアドホックに与えられるスケーリングパラメータを扱えることが重要である．**4.4** で述べたとおり，提案手法は事前計算を必要としないため，アドホックに与えれれるスケーリングパラメータを扱うことができる．提案手法のこの特徴を評価するために，ここでは複数のスケーリングパラメータを設定し評価を行った．図 4 に結果を示す．この実験ではデータセットとして

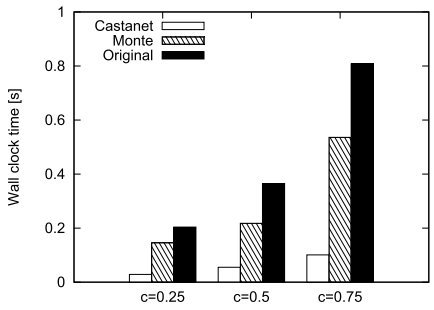

図 4　スケーリングパラメータと検索時間
Fig. 4　Efficiency versus scaling parameters.

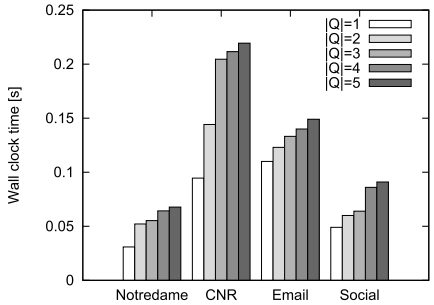

図 5　問い合わせノードの数と検索時間
Fig. 5　Efficiency versus number of query nodes.

Notredame を使用し，また $k = 10$ とした．さきの実験で示したとおり，行列分解に基づく手法は高速にアドホックに与えられるスケーリングパラメータは扱えないため，モンテカルロ法に基づく手法とのみ比較を行った．

図 4 に示すとおり，スケーリングパラメータが小さくなるほど，検索速度が向上することがわかる．これは提案手法ではスコアの上限値を式 (11) に示すとおり，下限値に $c^{i+1}W_{max}[u]\, p_i[R_i]$ を足すことで求めるが，ここで c が小さくなるほど上限値が小さくなるからである．またモンテカルロ法に基づく手法において c が大きくなるほどランダムウォークが確率的に長くなる．そのためモンテカルロ法に基づく手法は c が大きくなるほど検索時間が長くなる．

5.4　問い合わせノードの数に対する評価

4.3.2 で述べたとおり，提案手法は問い合わせノードからのホップ数が i 以内であるノードの集合 H_i を用いて部分グラフを計算し，計算対象となるノードを減らし高速な検索を可能にしている．ここでは問い合わせノードの数と検索時間について評価を行った．図 5 に結果を示す．なお図 5 において $|Q|$ は問い合わせノードの数を示している．

図 5 に示すとおり，提案手法は問い合わせノードの数が少なくなるほど高速に検索を行えることがわかる．これは定義 6 からわかるとおり，問い合わせノードの数はノードの集合 H_i の大きさに影響し，ノードの集合 H_i は問い合わせノードの数は部分グラフの大きさに影響するためである．問い合わせノードの数が少なくなるほど部分グラフが小さくなり，その結果提案手法における計算時間が減ることを確認した．

6. むすび

本論文では PPR に対して top-k のノードを高速に検索する問題に取り組んだ．本論文は Castanet を提案したが，この手法は下限値と上限値を用いて検索に不要なノードを枝狩りすることで高速な検索を実現している．実験的に提案手法と既存手法を比較し，提案手法がアドホックにグラフやスケーリングパラメータが与えられる状況において有効であることを確認した．

文　献

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” Technical Report 1999-66, Stanford InfoLab, Nov. 1999. http://ilpubs.stanford.edu:8090/422/

[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou, “Objectrank: Authority-based keyword search in databases,” VLDB, pp.564–575, 2004.

[3] G. Jeh and J. Widom, “Scaling personalized web search,” WWW, pp.271–279, 2003.

[4] A.N. Langville and C.D. Meyer, Google’s PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 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] K. Avrachenkov, N. Litvak, D. Nemirovsky, E. Smirnova, and M. Sokol, “Quick detection of top-k personalized pagerank lists,” WAW, pp.50–61, 2011.

[7] B. Bahmani, K. Chakrabarti, and D. Xin, “Fast personalized pagerank on mapreduce,” SIGMOD Conference, pp.973–984, 2011.

[8] B. Bahmani, A. Chowdhury, and A. Goel, “Fast incremental and personalized pagerank,” PVLDB, vol.4, no.3, pp.173–184, 2010.

[9] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, “Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments,” Internet Mathematics, vol.2, no.3, pp.333–358, 2005.

[10] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa, “Fast and exact top-k search for random walk with restart,” PVLDB, vol.5, no.5, pp.442–

453, 2012.
[11] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, “Efficient personalized pagerank with accuracy assurance,” KDD, pp.15–23, 2012.
[12] H. Tong, C. Faloutsos, and J.-Y. Pan, “Fast random walk with restart and its applications,” ICDM, pp.613–622, 2006.
[13] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, T. Mishima, and M. Onizuka, “Efficient ad-hoc search for personalized pagerank,” SIGMOD Conference, pp.445–456, 2013. http://doi.acm.org/10.1145/2463676.2463717
[14] H. Shiokawa, Y. Fujiwara, and M. Onizuka, “Fast algorithm for modularity-based graph clustering,” AAAI, 2013.
[15] D.A. Harville, Matrix Algebra From a Statistician’s Perspective, Springer, 2008.
[16] Y.Z. Guo, K. Ramamohanarao, and L.A.F. Park, “Personalized pagerank for web page prediction based on access time-length and frequency,” Web Intelligence, pp.687–690, 2007.
[17] B. Carterette, J. Allan, and R.K. Sitaraman, “Minimal test collections for retrieval evaluation,” SIGIR, pp.268–275, 2006.

（平成 26 年 7 月 26 日受付，11 月 20 日再受付，
27 年 2 月 4 日早期公開）

藤原　靖宏　（正員）

2003 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了．同年，日本電信電話株式会社入社．2011 年東京大学大学院情報理工学系研究科電子情報学専攻博士課程修了，2014 年ニューヨーク大学客員研究員．現在，日本電信電話株式会社 NTT ソフトウェアイノベーションセンタ特別研究員．博士（情報理工学）．グラフマイニングの研究開発に従事．第 27 回テレコムシステム技術賞，第 9 回上林奨励賞など受賞．情報処理学会，日本データベース学会各会員．

中辻　　真　（正員）

2001 年京大工学部数理工学科卒業．2003 年京大大学院情報学研究科システム科学専攻修士課程修了．2003 年日本電信電話株式会社入社．2010 年京大大学院情報学研究科社会情報学専攻博士課程修了（情報学）．2013 年から米国レンセラー工科大客員研究員．日本電信電話株式会社サービスエボリューション研究所所属．Semantic Web，情報推薦の研究に従事．人工知能学会会員．

塩川　浩昭

2009 年筑波大学第三学群情報学類卒業．2011 年同大学院システム情報工学研究科博士前期課程修了．同年，日本電信電話株式会社入社．現在，日本電信電話株式会社 NTT ソフトウェアイノベーションセンタ研究員，及び筑波大学大学院システム情報工学研究科博士後期課程在籍．大規模データ分析，分散並列処理の研究開発に従事．2013 年 DEIMForum 2013 最優秀論文賞及び優秀論文賞，2014 年 DEIM Forum2014 優秀論文賞，日本データベース学会平成 25 年度論文賞受賞．日本データベース学会会員．

三島　　健

1994 筑波大･第三学群･情報学類卒．1996 同大大学院工学研究科修士課程修了．同年，NTT 入社．電話交換システムの開発に従事．2010 東大工学研究科博士課程修了．博士（工学）．2010 年上林奨励賞受賞．日本データベース学会，情報処理学会，各会員．現在，データベースシステム，クラウドコンピューティングの研究に従事．

鬼塚　　真　（正員）

1991 年東京工業大学工学部情報工学科卒業．同年，日本電信電話株式会社入社．2000～2001 年ワシントン大学客員研究員，2010～2014 年日本電信電話株式会社特別研究員，2012～2014 年電気通信大学客員教授，現在，大阪大学大学院情報科学研究科教授．博士（工学）．大規模グラフデータの分散データ処理に関する研究開発に取り組んでいる．2004 年情報処理学会山下記念賞，2008 年データベース学会上林奨励賞など受賞．情報処理学会，日本データベース学会，ACM 各会員．