大規模グラフに対するObjectRankの高速化

佐藤　朋紀†　　塩川　浩昭††　　山口　祐人†††　　北川　博之††

† 筑波大学情報学群情報科学類　〒305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学計算科学研究センター　〒305–8577 茨城県つくば市天王台 1–1–1
††† 国立研究開発法人産業技術総合研究所　〒135–0064 東京都江東区青海 2–3–26
E-mail: †t.sato@kde.cs.tsukuba.ac.jp, ††{shiokawa,kitagawa}@cs.tsukuba.ac.jp,
†††yamaguchi.yuto@aist.go.jp

あらまし　ObjectRank はデータベース内のオブジェクトをグラフのノードとみなすことで，キーワード検索を実現するグラフ分析手法である．ObjectRank は複数種類のノードとエッジからなるグラフを扱うことができるため，多様なデータに対して適用することができる．しかしながら，ObjectRank はデータベース内に現れる全てのキーワードに対して各ノードが持つ重要度を繰り返し計算する必要があるため，大規模なグラフへの適用が難しいという問題がある．そこで本稿では，ObjectRank の高速化手法を提案する．提案手法では，計算対象のノード数を削減することで ObjectRank の高速化を図る．具体的には，重要度の計算過程において重要度の著しく低いノードを逐次的にグラフから枝刈りすることでノード数を削減する．提案手法は，計算過程で各ノードの重要度の取り得る上限値と下限値を計算することによって枝刈りをした場合に検索結果に与える影響が小さいノードを特定する．本稿では実データを用いた評価実験を行い，ObjectRank に対する提案手法の有効性を評価する．

キーワード　ObjectRank，グラフマイニング，情報検索

1. 序　　論

ソーシャルメディアやモバイル機器の普及に伴い日々大量のデータが生成されており，データから新たに知識を獲得するデータ分析の需要が増加している．特に，データ間の関係性を表現するグラフと呼ばれるデータ構造に着目したグラフ分析は，データ間の関係性に隠れた知識を獲得できる分析技術であり，推薦システムや SNS(Social Networking Service) といったサービスで広く用いられている [1–3].

グラフ分析技術のひとつに，ObjectRank [4,5] がある．ObjectRank は，PageRank [6] を拡張することでデータベース内のオブジェクトに対するキーワード検索を実現する手法である．データベース内のオブジェクトをグラフに見立てることによって PageRank のリンク解析手法を適用し，各オブジェクトの重要度を評価する．PageRank とは異なり，ObjectRank は複数の種類のノードとエッジからなるグラフを扱うことができるため，多様なデータに対して適用可能である [7,8]．ObjectRank の実用的な応用例の一つに，TURank [9] がある．TURank は，マイクロブログサービスのひとつである Twitter (注1) に ObjectRank を適用することで，Twitter のユーザの評価とランク付けを行う手法である．TURank は，Twitter におけるユーザ間の様々な関係性をグラフで表現することによって ObjectRank の重要度評価手法を適用している．

ObjectRank は，クエリが与えられると行列ベクトル積の繰り返し演算によりグラフ全体のノードの重要度を評価する必要があるため，グラフ内のノード数を N，エッジ数を M，演算の繰り返し回数を T とすると，重要度評価の計算量は $O((N+M)T)$ となる．ゆえに，ノード数が大きいときクエリ応答時間は増加する．一般的に，ObjectRank は応答時間の高速化のために，事前にデータベース内に現れる全てのキーワードに対して各ノードの重要度を評価し索引付けする．しかし，データベース内のキーワード数を K とすると索引構築の計算量は $O((N+M)TK)$ となり，大規模なグラフを対象としたとき K，N，M が爆発的に増加し，索引構築に膨大な時間を必要とする．したがって，ObjectRank の高速化は重要な研究課題となっている．

1.1 本研究の貢献

本研究では，ObjectRank の重要度評価の高速化手法を提案する．提案手法では，計算対象のノード数 N を削減することで ObjectRank の高速化を図る．具体的には，重要度評価の計算過程で重要度の著しく低いノードを逐次的に枝刈りする．提案手法は，各ノードの重要度の取り得る上限値と下限値を計算することによって枝刈りした場合に検索結果に与える影響の少ないノードを特定することができるため，高い精度で結果を近似することができる．この手法により重要度評価の繰り返し計算を進めるにつれて計算対象のノードが減少するため，従来の ObjectRank と比較して重要度評価を高速に行うことができる．

本研究の貢献は下記の通りである．

- **高速性:** 提案手法は従来の ObjectRank と比較して重要度評価を高速に行うことができる．提案手法は 100 万のノードと 500 万のエッジを持つグラフに対し，約 1 秒で重要度評価を行うことができる（4.3.1 節）．
- **近似精度:** 提案手法は高い精度で ObjectRank の上位 k 件の結果を近似することができる．提案手法は上位 100 件の結

(注1)：http://twitter.com

果を 90 % 以上の精度で近似することができる．(4.3.2 節)．

上述の通り，ObjectRank は計算コストが膨大であり，大規模グラフに適用することは難しい．しかし，提案手法は逐次的なノードの枝刈りを行うことで計算対象のノード数を削減し，重要度評価を高速に行うことができる．実データを用いた評価実験では，90 % 以上の精度で ObjectRank と比較して約 8 倍重要度評価を高速に行うことができることを示した．

本稿の構成は次の通りである．まず，2 節で前提となる知識について説明し，3 節で提案手法について述べる．4 節で評価実験について説明し，5 節で関連研究を紹介する．そして最後に 6 節でまとめと今後の課題について述べる．

2. 前提知識

表 1 に本稿が用いる記号の定義を示す．

表 1　本稿が用いる記号の定義

記号	定義
G	計算対象のグラフ
V	G に含まれるノードの集合
E	G に含まれるエッジの集合
N	グラフのノード数
M	グラフのエッジ数
K	データベース内のキーワード数
$\boldsymbol{A}$	$N \times N$ の遷移行列
d	ダンピングファクタ
ϵ	閾値
$\boldsymbol{r}$	各ノードの重要度を並べた $N \times 1$ のベクトル

ObjectRank [4,5] はデータベースに対するキーワード検索を実現する手法である．2.1 節で ObjectRank の概要について，2.2 節で ObjectRank の重要度評価について，2.3 節で ObjectRank の問題点についてそれぞれ述べる．

2.1 ObjectRank の概要

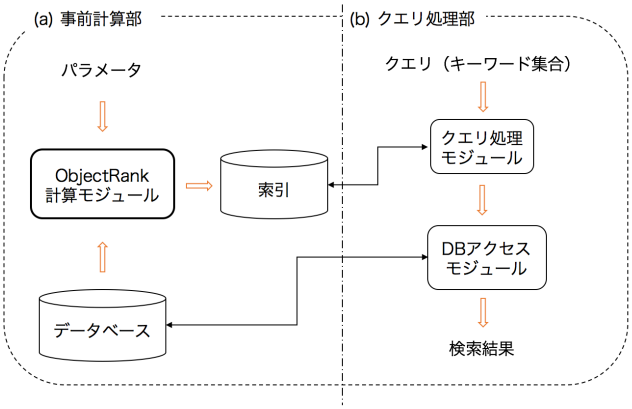

図 1　ObjectRank のシステム

ObjectRank のシステムの概要を図 1 に示す．ObjectRank のシステムは (a) 事前計算部と (b) クエリ処理部の二つに分けることができる．(a) 事前計算部では，各種パラメータを引数にとり ObjectRank 計算モジュールにおいてデータベース内の各オブジェクトの重要度を計算し，結果を索引付けする．(b) クエリ処理部ではユーザから与えられたクエリと索引に基づいてオブジェクトのランキングを作成し，重要度が上位 k 件のオブジェクトをユーザに返す．本研究は事前計算部における重要度評価の高速化を図るため，以降では ObjectRank の重要度評価について説明を行う．

2.2 ObjectRank の重要度評価

2.2.1 データモデル

ObjectRank はデータベース内のオブジェクトをラベル付き有向グラフとして表現する．まず，グラフのノードとエッジの種類，及びエッジの重みを定義した *Authority Transfer Schema Graph* を作成する（以降，*Schema Graph* と呼ぶ）．各種ノードとエッジはラベルが付与され，ラベルによって種類が区別される．各エッジの重みはそのエッジによって遷移する重要度の割合を示す．ただし，重みは 0 以上 1 以下の値をとり，一つのノードから出るエッジの重みの総和は 1 以下に設定する必要がある．図 2 に文献データベースを表現した *Schema Graph* の例を示す．図 2 の例では，“Conference” や “Year” といったラベルが付与された 4 種類のノードと，それらをつなぐ 8 種類のエッジが存在する．

次に，*Schema Graph* に基づいて対象のデータから *Authority Transfer Data Graph* を構築する（以降，*Data Graph* と呼ぶ）．*Data Graph* の各ノードは “VLDB” や “2004” といったキーワードを保持する．*Data Graph* のエッジの重みは，*Schema Graph* で定義した重みをエッジの元ノードの出次数で割った値となる．ただし，出自数は同じ種類のエッジの本数のみを数えたものとする．図 3 に 図 2 に基づいて作成された *Data Graph* の例を示す．“Balmin, A.” ノードから “ObjectRank” ノードには重みが 0.1 であるエッジが張られている．これは，*Schema Graph* の “Author” から “Paper” へのエッジの重みである 0.2 を，“Balmin, A.” の出次数 2 で割った値である．

図 2　文献データベースの *Schema Graph*

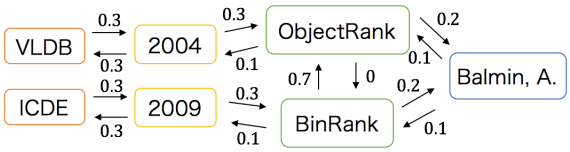

図 3　図 2 に基づいて作成された *Data Graph* の例

2.2.2 重要度の計算方法

ObjectRank は，*Data Graph* に対して重要度評価を行う．*Data Graph* の遷移行列を $\boldsymbol{A}$ とする．*Data Graph* のノード数を N とすると $\boldsymbol{A}$ は $N \times N$ の行列であり，(i,j) 要素にはノード v_j から v_i にエッジ e_{ji} が存在する場合は e_{ji} の重みを，それ以外の場合は 0 を格納する．このとき，あるキーワード t に対する G の各ノードの重要度を並べたベクトル $\boldsymbol{r} = [r(v_1), r(v_2), \ldots, r(v_n)]^T$ は以下の式で得られる．

$$\boldsymbol{r} = d\boldsymbol{A}\boldsymbol{r} + (1-d)\boldsymbol{q} \tag{1}$$

ただし，d $(0 < d < 1)$ はダンピングファクタ，$BS(t)$ をキーワード t を含むノードの集合としたとき，$\boldsymbol{q}$ は n 次元のクエリベクトルであり，以下のように求める．

$$q(v) = \begin{cases} 1/|BS(t)| & (v \in BS(t)) \\ 0 & (otherwise) \end{cases}$$

ObjectRank は，べき乗法を用いて式 (1) を解く．すなわち，任意の初期値を与えて以下の式を繰り返し計算する．

$$\boldsymbol{r}^{i+1} \leftarrow d\boldsymbol{A}\boldsymbol{r}^i + (1-d)\boldsymbol{q} \tag{2}$$

$\boldsymbol{r}^i$ は i 回目の繰り返し計算によって求めた重要度のベクトルである．$\boldsymbol{r}$ が収束したとき繰り返し計算を終了する．

2.2.3 索引構築

ObjectRank はクエリ応答の効率化のために，クエリ処理時に重要度評価を行う代わりに事前に索引構築を行う．まずデータベースに現れるキーワード t について各ノードの重要度 $\boldsymbol{r}_t$ を計算し，ノードの id $id(u)$ と重要度 $r_t(u)$ のペア $< id(u), r_t(u) >$ を $r_t(u)$ の降順で並べたリストを作成する．次に，重要度 $r_t(u)$ が任意の閾値 ϵ を下回るノードのペアをリストから取り除き，リストを索引に格納する．最後に，上述の処理をデータベースに現れる全てのキーワードについて実行することで索引を構築する．

2.3 ObjectRank の問題点

ObjectRank は，べき乗法を用いて各ノードの重要度が収束するまで式 (1) を繰り返し計算する必要があるため，重要度評価の計算量は $O((N+M)T)$ となる．索引構築時にはこの計算をデータベース内に現れる全てのキーワードに対して行う必要があり，索引構築の計算量は $O((N+M)TK)$ となる．大規模グラフを対象とした際には N，M，および K が増加するため，計算コストが爆発的に増大するという問題がある．

3. 提案手法

本節では，提案手法について説明を行う．3.1 節で概要を述べ，3.2 節で提案手法の詳細を説明する．

3.1 概　　要

提案手法は，計算対象とするノード数 N を削減することによって ObjectRank の重要度評価の高速化を図る．2.2.3 節で述べた通り，ObjectRank は重要度が閾値 ϵ より低いノードについてはノードの id と重要度のペア $< id(u), r(u) >$ の索引への格納は省略する．すなわち，重要度の著しく低いノードは検索結果に影響を持ちにくいという性質がある．そこで提案手法では，重要度評価における繰り返し計算の過程で検索結果に影響を持ちにくいノードを特定し，計算対象から逐次的に枝刈りすることによってノード数を削減する．

3.2 提案手法の詳細

提案手法では，枝刈り可能なノードを特定するために各ノードが取り得る重要度 $\boldsymbol{r}$ の上限値 $\overline{\boldsymbol{r}}$ と下限値 $\underline{\boldsymbol{r}}$ を推定する．ノード u_i の上限値 $\overline{r}(u_i)$ が任意の閾値 ϵ を下回ったとき，u_i の最終的な重要度も同様に ϵ を下回ることが保証されるため，重要度の計算対象から枝刈りする．下限値 $\underline{\boldsymbol{r}}$ は，繰り返し計算の過程で上限値 $\overline{\boldsymbol{r}}$ を効率的に更新するために用いる．

3.2.1 上限値と下限値

重要度の下限値 $\underline{\boldsymbol{r}}$ と上限値 $\overline{\boldsymbol{r}}$ を以下に定義する．

［定義 3.1］（下限値） i 番目の繰り返し計算におけるノード v の下限値は次のように計算する．

$$\underline{r}^i(v) = (1-d)\sum_{j=0}^{i} d^j p_j(v) \tag{3}$$

［定義 3.2］（上限値） i 番目の繰り返し計算におけるノード v の上限値は次のように計算する．

$$\overline{r}^i(v) = (1-d)\sum_{j=0}^{i} d^j p_j(v) + d^{i+1} p_i(v) + \frac{d^{i+1}}{(1-d)}\Delta_i \overline{A}(v) \tag{4}$$

$\boldsymbol{p}_i$ は長さ i のランダムウォークの確率を表す N 次元ベクトルであり，$\boldsymbol{p}_i = \boldsymbol{A}^i\boldsymbol{q}$ で計算する．ただし，$i = 0$ であるとき $\boldsymbol{p}_i = \boldsymbol{q}$ とする．また，Δ_i は次のように計算する．

$$\Delta_i = \begin{cases} 1 & (i = 0) \\ \sum_{u \in V_0} \Delta_i(v) & (i \neq 0) \end{cases}$$

ただし，G_i を i 番目の繰り返し計算における部分グラフ，G_0 を元のグラフとしたとき，V_0 は G_0 のノード集合を表す．$\Delta_i(v)$ は $\Delta_i(v) = max\{p_i(v) - p_{i-1}(v), 0\}$ によって計算する．$\overline{\boldsymbol{A}}$ はエッジの最大の重みを格納した N 次元ベクトルであり，$\overline{A}(v) = max\{A(v,u) : u \in G_i\}$ となる．

［補題 3.1］（下限値の性質） i 番目の繰り返し計算において，下限値 $\underline{r}^i(v)$ は次の性質を満たす．

$$\underline{r}^i(v) \leqq r(v) \tag{5}$$

〈証明〉式 (2) より，

$$\begin{aligned} \boldsymbol{r}^i &= d\boldsymbol{A}\boldsymbol{r}^{i-1} + (1-d)\boldsymbol{q} \\ &= d^2\boldsymbol{A}^2\boldsymbol{r}^{i-2} + (1-d)(d\boldsymbol{A}\boldsymbol{q} + \boldsymbol{q}) \\ &= d^i\boldsymbol{A}^i\boldsymbol{r}^0 + (1-d)\{d^{i-1}\boldsymbol{A}^{i-1}\boldsymbol{q} + d^{i-2}\boldsymbol{A}^{i-2}\boldsymbol{q} + \cdots + \boldsymbol{q}\} \\ &= d^i\boldsymbol{A}^i\boldsymbol{r}^0 + (1-d)\sum_{j=0}^{i-1} d^j\boldsymbol{p}_j \end{aligned}$$

最終的な重要度のベクトルは $\boldsymbol{r}$ の収束値であるため，$\boldsymbol{r} = \boldsymbol{r}^\infty$ となる．$0 < d < 1$ かつ $\boldsymbol{A}^\infty$ の各要素は 0 以上 1 以下の値をとるため，

$$\begin{aligned} \boldsymbol{r} &= d^\infty\boldsymbol{A}^\infty\boldsymbol{r}^0 + (1-d)\sum_{j=0}^{\infty} d^j\boldsymbol{s}_j \\ &= (1-d)\sum_{j=0}^{\infty} d^j\boldsymbol{s}_j \end{aligned} \tag{6}$$

が成り立つ．したがって，

$$\boldsymbol{r} = (1-d)\sum_{j=0}^{\infty} d^j \boldsymbol{s}_j \tag{7}$$

$$\geqq (1-d)\sum_{j=0}^{i} d^j \boldsymbol{s}_j \tag{8}$$

が成り立つため，補題 3.1 の式 (5) が得られる． □

［補題 3.2］（上限値の性質） i 番目の繰り返し計算において，上限値 $\overline{r}^i(v)$ は次の性質を満たす．

$$\overline{r}^i(v) > r(v) \tag{9}$$

〈証明〉付録 1. を参照． □

重要度評価の際の繰り返し計算において，下限値 $\underline{\boldsymbol{r}}$ と上限値 $\overline{\boldsymbol{r}}$ は次のように逐次的に計算することができる．

［補題 3.3］（下限値と上限値の逐次的な計算） 下限値 $\underline{\boldsymbol{r}}_i$ と上限値 $\overline{\boldsymbol{r}}_i$ の逐次的な計算は以下のように行う．また，次式は $O(1)$ で計算可能である．

$$\underline{r}^i(v) = \begin{cases} (1-d)q(v) & (i=0) \\ \underline{r}^{i-1} + (1-d)d^i p_i(v) & (i \neq 0) \end{cases} \tag{10}$$

$$\overline{r}^i(v) = \begin{cases} q(v) + \frac{d}{(1-d)}\overline{A}(v) & (i=0) \\ \underline{r}^{i-1} + d^i p_i(v) + \frac{d^{i+1}}{(1-d)}\Delta_i \overline{A}(v) & (i \neq 0) \end{cases} \tag{11}$$

〈証明〉$i = 0$ であるとき，定義 1 より $\underline{r}^i(v) = (1-d)\sum_{j=0}^{i} d^j p_j(v) = (1-d)p_0(v) = (1-d)q(v)$．また，定義 2 より $\overline{r}^i(v) = (1-d)q(v) + dq(v) + \frac{d}{(1-d)}\overline{A}(v)$ となり，$\overline{r}^i(v) = q(v) + \frac{d}{(1-d)}\overline{A}(v)$ が成り立つ．このとき，d，$q(v)$，$\overline{A}(v)$ は定数であるため，$\overline{r}^i(v)$ と $\underline{r}^i(v)$ は $O(1)$ で計算可能である．

$i \neq 0$ であるとき，定義 1 より $\underline{r}^i(v) - \underline{r}^{i-1}(v) = (1-d)\sum_{j=0}^{i} d^j p_j(v) - (1-d)\sum_{j=0}^{i-1} d^j p_j(v) = (1-d)d^i p_i(v)$ となり，$\underline{r}^i(v) = \underline{r}^{i-1} + (1-d)d^i p_i(v)$．また，定義 1，2 より，$\overline{r}^i(v) - \underline{r}^{i-1}(v) = d^i p_i(v) + \frac{d^{i+1}}{(1-d)}\Delta_i \overline{A}(v)$ となり，$\overline{r}^i(v) = \underline{r}^{i-1}(v) + d^i p_i(v) + \frac{d^{i+1}}{(1-d)}\Delta_i \overline{A}(v)$ が成り立つ．このとき，$\underline{r}^{i-1}(v)$ は $i-1$ 番目の繰り返し計算時においてあらかじめ計算されており，d，$p_i(v)$，$\overline{A}(v)$ は定数であるため，$\overline{r}^i(v)$ と $\underline{r}^i(v)$ は $O(1)$ で計算可能である． □

3.3 アルゴリズム

本節では，提案手法のアルゴリズムについて述べる．Algorithm 1 に提案手法の疑似コードを示す．Algorithm 1 は，グラフ G，閾値 ϵ，最大反復回数 τ を入力に受け取り，枝刈りされたノードを除いた各ノードの重要度を計算する．ただし，閾値 ϵ は計算対象のノード数で割った値を使用する（4 行目）．これは，各ノードの重要度の総和は 1 となり，ノード数が大きいときはノードの重要度や上限値，下限値は小さい値を取ってしまうためである．

i ($i = 0, 1, \ldots$) 番目の繰り返し計算において，各ノードの重要度 $r_i(v)$，上限値 $\overline{r}^i(v)$，下限値 $\underline{r}^i(v)$ を計算する（6，7 行目）．このとき、定義 **??** より上限値と下限値は $O(1)$ で計算することができる．次に，得られた上限値が ϵ を下回ったノード u と u に隣接するエッジをグラフ G_i から取り除き，新たな部分グラフ G_{i+1} を構築する（12 行目）．i 番目の繰り返し計算において，G_i に含まれる全てのノードの重要度が収束したとき，または繰り返し計算回数がユーザの設定した上限 τ に達したときにアルゴリズムを終了する．

Algorithm 1 提案手法のアルゴリズム

Input: G, 元のグラフ; ϵ, 閾値; τ, 最大反復回数
Output: 枝刈りされたものを除いた各ノードの重要度

```
1:  i ← 0
2:  G_i ← G
3:  while i < τ do
4:      ε ← ε/G_i のノード数
5:      for each G_i に含まれるノード v に対して do
6:          式 (2 より重要度 r_i(v) を計算
7:          式 (10)，(11) より上限値 r̄^i(v)，下限値 r̲^i(v) を計算
8:      end for
9:      if G_i の全てのノードの重要度が収束 then
10:         アルゴリズムを終了する
11:     end if
12:     G_i から上限値が ε を下回るノードを取り除き G_{i+1} を構築
13:     i ← i + 1
14: end while
```

4. 評価実験

本節では，実データに対して提案手法および ObjectRank を実行し，実行速度と上位 k 件の結果の近似精度の観点で提案手法の有効性を検証する．実験環境の詳細は表 2 に示す．

表 2 実験環境

CPU	Intel Xeon (E5-1620 v4) 3.5GHz
メモリ	128GB
OS	CentOS (7.3.1611 (Core))
言語	C++11 (gcc 4.8.5)

4.1 データセット

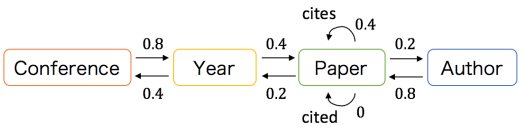

図 4 *Schema Graph* 2

データセットは AMiner (注2) が公開している文献データベース [10–14] を用いた．このデータセットには文献のタイトル，発表年，発表場所，著者名，引用している文献のリストの情報が含まれている．*Schema Graph* で定義したエッジの重みの実験結果への影響も加味するために，グラフは 2 つの *Schema Graph* に基づいて作成した．1 つは図 2 の *Schema Graph* に基づいて構築し（以降，*Schema Graph* 1 とする），2 つ目は各ノードの出エッジの重みの総和が全て等しくなるように設定し

(注2)：http://aminer.org/

た図 4 の *Schema Graph* に基づいて作成した（以降，*Schema Graph* 2 とする）．*Schema Graph* 2 は *Schema Graph* 1 と比較して，全ての種類のノードが高い重要度を持ちやすいように設定されていると考えることができる．作成したグラフのノード数とエッジ数の詳細は表 3 に示す．

表 3　ノード数とエッジ数の詳細

Conference のノード数	12,609
Year のノード数	67
Paper のノード数	629814
Author のノード数	595776
Conference，Year 間のエッジ数	48
Year，Paper 間のエッジ数	1,259,628
Paper，Paper 間のエッジ数	1,265,502
Paper，Author 間のエッジ数	2,624,116
総ノード数	1,238,266
総エッジ数	5,149,294

4.2 実験方法

本実験は，単一キーワードについて重要度評価を行った際の実行時間と上位 k 件の近似精度を計測した．本稿では，“Paper” および “Conference” ラベルのノードから “PageRank”，“VLDB” というキーワードを選択し，これらに対する重要度をそれぞれ計算した．すなわち，以下の 4 通りの条件で実験を行った．

- (条件 1) *Schema Graph*: 1，キーワード: “PageRank”
- (条件 2) *Schema Graph*: 2，キーワード: “PageRank”
- (条件 3) *Schema Graph*: 1，キーワード: “VLDB”
- (条件 4) *Schema Graph*: 2，キーワード: “VLDB”

このとき，ダンピングファクタ d は 0.85，最大計算回数 τ は 100 回に固定し，閾値 ϵ の値を変更して実行した．近似精度の指標は以下を用いた．

$$Prec(t,k) = \frac{|ProSet(t,k) \cap ORSet(t,k)|}{k} \qquad (12)$$

$ProSet(t,k)$ と $ORSet(t,k)$ は，キーワード t に対して提案手法と ObjectRank の重要度評価をそれぞれ実行し，得られた上位 k のノードの集合である．$Prec(t,k)$ は 0 以上 1 以下の値を取り，近似精度が良くなるにつれて 1 に近い値をとる．

4.3 実験結果

4.3.1 実行速度

様々な閾値 ϵ の値に対する ObjectRank と提案手法の実行時間をそれぞれ計測した．条件 1，条件 2，条件 3，条件 4 について実行した結果をそれぞれ図 5，図 6，図 7，図 8 に示す．実験結果より，いずれの条件の場合でも閾値の値が小さい場合は提案手法の方が遅くなってしまうということがわかる．閾値の値が小さい場合は枝刈りが行われにくく，上限値の計算によるオーバーヘッドが影響したためであると考えられる．一方，閾値の値が大きい場合は枝刈りが上手く行われ，提案手法の方が高速に結果を得られることがわかる．条件 1（図 5）と条件 2（図 6）の間に大きな差は見られないが，条件 3（図 7）と条件 4（図 8）を比較すると，条件 3 の方がより小さな閾値の値で提案手法が ObjectRank よりも高速になっていることがわかる．これは，条件 4 は *Schema Graph* の重みを図 2 から図 4 に変更したことで検索結果へと影響を与えにくいノードが減少し，枝刈りが行われにくくなったためであると考えられる．

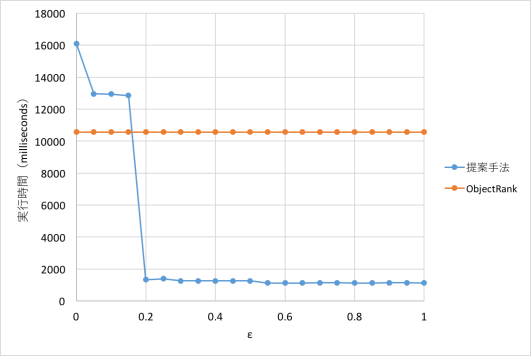

図 5　実行時間（条件 1）

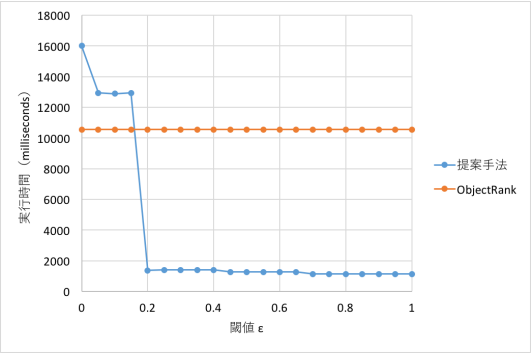

図 6　実行時間（条件 2）

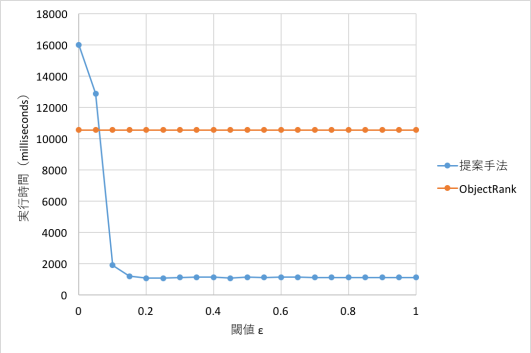

図 7　実行時間（条件 3）

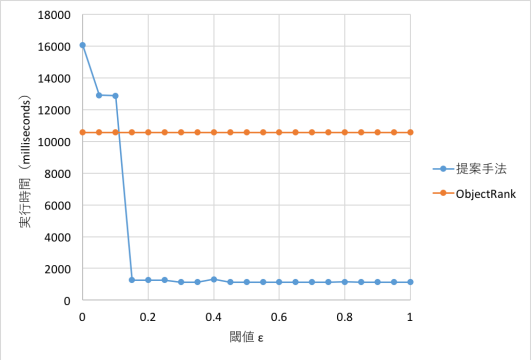

図 8　実行時間（条件 4）

4.3.2 近似精度

ObjectRank によって得られた上位 k 件の結果を真値とし，式 (12) を用いて様々な閾値 ϵ に対する提案手法の上位 k 件の近似精度を計測した．条件 1，条件 2，条件 3，条件 4 について実行した結果をそれぞれ図 9，図 10，図 11，図 12 に示す．実験結果より，いずれの条件の場合でも上位 50 件程度であれば 90 % 以上の高い精度で結果を得られることが示された．提案手法は計算過程において重要度の上限値と下限値を計算することによって検索結果に影響を持ちにくいノードの特定を目指したが，実験結果より枝刈り対象のノードが適切に選択されたことがわかる．また，条件 1（図 9）と条件 2（図 10，および条件 3（図 11）と条件 4（図 12）を比較すると，*Schema Graph* 1 を用いた場合よりも *Schema Graph* 2 を用いた場合の方が近似精度が低くなる傾向が見られた．これは，*Schema Graph* の重みを図 2 から図 4 に変更したことで，検索結果へと影響を与えにくいノードが減少したためであると考えられる．

実行時間および近似精度の実験結果より，提案手法は閾値が小さい場合は実行時間が長く，近似精度が高くなった．一方で，閾値が大きい場合は実行時間が短く，近似精度は比較的低くなったため，実行時間と近似精度の間にはトレードオフの関係が存在する．最適な閾値の値の選択は今後の課題としてあげられる．

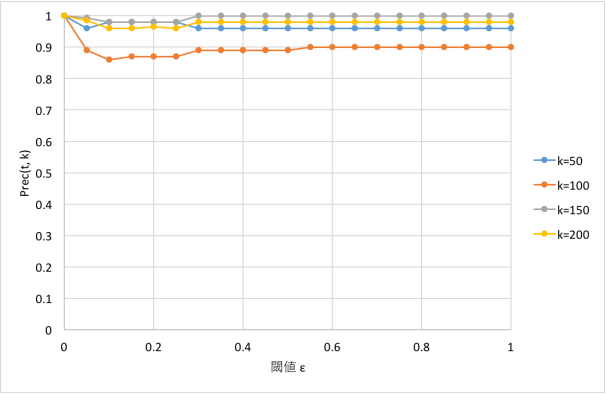

図 9　近似精度（条件 1）

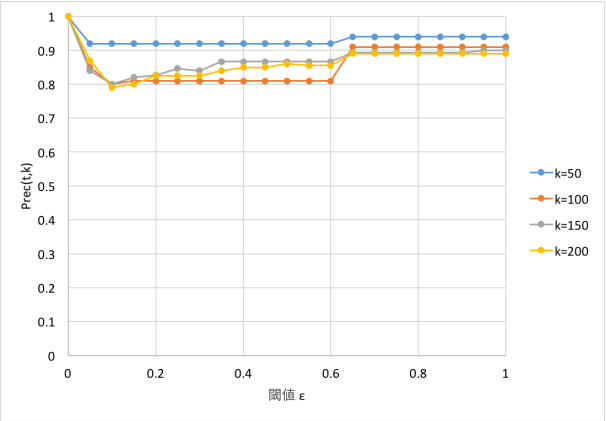

図 10　近似精度（条件 2）

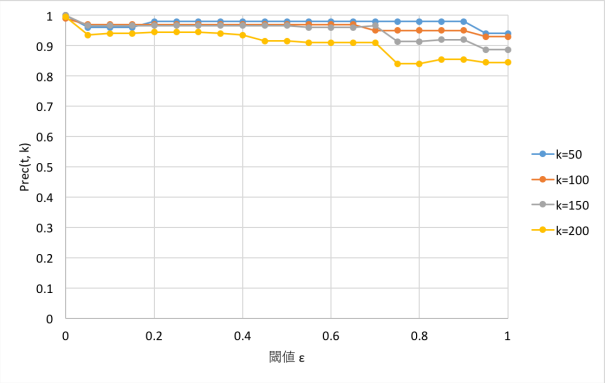

図 11　近似精度（条件 3）

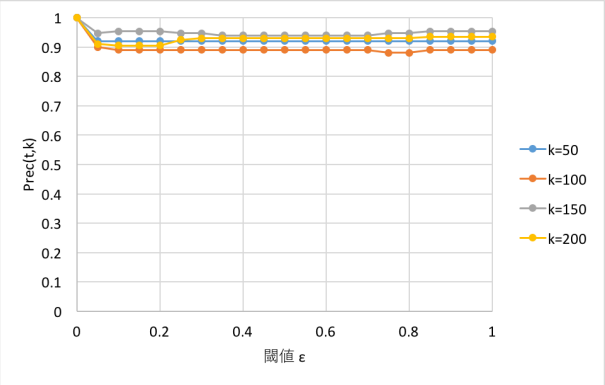

図 12　近似精度（条件 4）

5. 関連研究

5.1 PageRank に対する関連研究

Monte Carlo 法 [15–17] は，べき乗法による繰り返し計算を行う代わりに，PageRank におけるランダムサーファーモデルに基づいて重要度が遷移する様子をランダムに試行する手法である．ランダムサーファーモデルとは，Web のランダムなページから遷移を開始し，ランダムなハイパーリンクを選択してページの遷移を繰り返すモデルである．Monte Carlo 法では，ランダムサーファーの遷移を複数回試行して終着点を記録する．あるページの重要度は，そのページが終着点となった回数に応じて決定される．しかし，[16] で述べられている通り高い精度で結果を近似するためには試行回数を増やす必要がある．

Fujiwara ら [18] は，繰り返し計算の過程で解になり得ないノードを逐次的に枝刈りし，PageRank の Top-k 検索を高速化する手法 F-Rank を提案した．F-Rank では，繰り返し計算の各ステップにおいて各ノードの重要度の上限値と下限値の推定値を用いて Top-k 検索の結果になり得ないノードを特定し，グラフから枝刈りする．グラフからノードの枝刈りを行うが，検索結果は理論的に正しいことが保障されている．しかし，F-Rank は遷移行列の列が正規化される PageRank の特性を利用しているため，ObjectRank には適用できない．

5.2 ObjectRank に対する高速化手法

Hwang ら [19] は，元のグラフから比較的ノード数の少ない部分グラフを構築することで ObjectRank を高速化する手

法 BinRank を提案した．計算対象のキーワードをクラスタリングし，クラスタに含まれるキーワードとの関連度が著しく低いノードを取り除いた部分グラフを，クラスタごとに構築し索引に格納する．クエリ処理時には，キーワードが含まれるクラスタから作成された部分グラフのみを対象に重要度評価を行う．しかし BinRank は，キーワードのクラスタごとに部分グラフを構築し索引付けするため，対象のグラフサイズが大きいとき索引サイズが膨大となる．また，クエリ処理時にべき乗法による繰り返し計算を行うためクエリ応答に時間がかかってしまい，依然として計算コストが問題となる．

6. 結　　論

本稿では大規模グラフに対する ObjectRank の高速化手法を提案した．提案手法では計算対象とするノード数 N を削減することによって ObjectRank の高速化を図った．提案手法は重要度評価の繰り返し計算の過程において重要度の著しく低いノードを逐次的にグラフから枝刈りすることでノード数を削減する．提案手法は，計算過程で各ノードの重要度の取り得る上限値と下限値を計算することによって検索結果に影響を持ちにくいノードを特定することができるため，高い精度で結果を近似することができる．実データを用いた評価実験では，提案手法は ObectRank と比較して高速かつ精度よく重要度評価を行うことができることを示した．

今後の課題として，計算対象のキーワード数 K の削減による ObjectRank の高速化手法の提案があげられる．2.3 節で述べた通り ObjectRank の索引構築の計算量は $O((N+M)TK)$ となるため，キーワード数 K の削減により ObjectRank の索引構築の高速化が期待できる．

謝　　辞

本研究成果は，JSPS 科研費 (26280037,16H07410) の助成を受けたものである．

文　　献

[1] M. Gori and A. Pucci, “ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines,” in *Proc. IJCAI*, pp. 2766–2771, 2007.
[2] B. Bahmani, A. Chowdhury, and A. Goel, “Fast Incremental and Personalized PageRank,” *Proc. VLDB*, vol. 4, pp. 173–184, Dec. 2010.
[3] S. E. Schaeffer, “Graph Clustering,” *Computer Science Review*, vol. 1, pp. 27–64, Aug. 2007.
[4] A. Balmin, V. Hristidis, and Y. Papakonstantinou, “ObjectRank: Authority-Based Keyword Search in Databases,” in *Proc. VLDB*, pp. 564–575, 2004.
[5] V. Hristidis, H. Hwang, and Y. Papakonstantinou, “Authority-Based Keyword Search in Databases,” *ACM Trans. Database Syst.*, vol. 33, Mar. 2008.
[6] S. Brin and L. Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine,” in *Proc. WWW*, pp. 107–117, 1998.
[7] V. Hristidis, Y. Wu, and L. Raschid, “Efficient Ranking on Entity Graphs with Personalized Relationships,” *TKDE*, vol. 26, pp. 850–863, Apr. 2014.
[8] W. Xie, D. Bindel, A. Demers, and J. Gehrke, “Edge-Weighted Personalized Pagerank: Breaking A Decade-Old Performance Barrier,” in *Proc. KDD*, pp. 1325–1334, 2015.
[9] Y. Yamaguchi, T. Takahashi, T. Amagasa, and H. Kitagawa, “TURank: Twitter User Ranking Based on User-Tweet Graph Analysis,” in *Proc. WISE*, pp. 240–253, 2010.
[10] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “Arnetminer: Extraction and mining of academic social networks,” in *Proc. KDD’08*, pp. 990–998, 2008.
[11] J. Tang, L. Yao, D. Zhang, and J. Zhang, “A Combination Approach to Web User Profiling,” *TKDD*, vol. 5, pp. 2:1–2:44, Dec. 2010.
[12] J. Tang, A. C. M. Fong, B. Wang, and J. Zhang, “A Unified Probabilistic Framework for Name Disambiguation in Digital Library,” *TKDE*, vol. 24, pp. 975–987, June 2012.
[13] J. Tang, J. Zhang, R. Jin, Z. Yang, K. Cai, L. Zhang, and Z. Su, “Topic Level Expertise Search over Heterogeneous Networks,” *Machine Learning Journal*, vol. 82, pp. 211–237, Feb. 2011.
[14] J. Tang, D. Zhang, and L. Yao, “Social Network Extraction of Academic Researchers,” in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM ’07, pp. 292–301, 2007.
[15] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, “Monte Carlo Methods in PageRank Computation: When One Iteration is Sufficient,” *SIAM J. Numer. Anal.*, vol. 45, pp. 890–904, Feb. 2007.
[16] D. Fogaras, B. Rcz, K. Csalogny, and T. Sarls, “Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments,” *Internet Mathematics*, vol. 2, no. 3, pp. 333–358, 2005.
[17] S. Chakrabarti, “Dynamic Personalized Pagerank in Entity-Relation Graphs,” in *Proc. WWW*, pp. 571–580, 2007.
[18] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, T. Mishima, and M. Onizuka, “Fast and Exact Top-k Algorithm for PageRank,” in *Proc. AAAI 2013*, pp. 1106–1112.
[19] H. Hwang, A. Balmin, B. Reinwald, and E. Nijkamp, “BinRank: Scaling Dynamic Authority-Based Search Using Materialized SubGraphs,” in *Proc. ICDE*, pp. 66–77, 2009.

付　　録

1. 補題 3.2 の証明

〈証明〉式 (6) より，

$$\begin{aligned} r(v) &= (1-d)\sum_{j=0}^{\infty} d^j p_j(v) \\ &= (1-d)\sum_{j=0}^{i} d^j p_j(v) + (1-d)\sum_{j=1}^{\infty} d^{i+j} p_{i+j}(v) \end{aligned} \tag{A·1}$$

まず，

$$p_{i+j}(v) \leqq p_i(v) + j\Delta_i \overline{A}(v) \tag{A·2}$$

が成り立つことを示す．$H_j(v)$ を j 回エッジをたどることでノード v に到達可能なノードの集合，R_i を V_i に含まれるノードに到達可能なノードの集合とする．$\boldsymbol{p}_{i+j} - \boldsymbol{p}_{i+j-1} = \boldsymbol{A}^{i+j}\boldsymbol{q} - \boldsymbol{A}^{i+j-1}\boldsymbol{q} = \boldsymbol{A}\boldsymbol{A}^{j-1}(\boldsymbol{p}_i - \boldsymbol{p}_{i-1})$ より，

$$\begin{aligned} & p_{i+j}(v) - p_{i+j-1}(v) \\ &= \textstyle\sum_{u\in H_1(v)} \sum_{w\in H_{j-1}(u)} A(v,u)A^{j-1}(u,w)(p_i(w) - p_{i-1}(w)) \end{aligned}$$

$$\leqq \sum_{w\in H_{j-1}(u)} \sum_{u\in H_1(v)} \overline{A}(v) A^{j-1}(u,w)\Delta_i(w)$$

$$\leqq \overline{A}(v) \sum_{w\in R_i} \Delta_i(w) (\sum_{u\in H_1(v)} A^{j-1}(u,w)) \qquad \text{(A·3)}$$

となる．ここで，α_{max} を Schema Graph 内のノードの出エッジの重みの総和の最大値とする．図 2 の例の場合，“Paper” の出エッジの重みの総和は 1.0 であり，Schema Graph 内で最大となる．したがって，$\alpha_{max} = 1.0$ となる．

$$\sum_{u\in H_1(v)} A^{j-1}(u,w) \leqq \alpha_{max}^{j-1} \qquad \text{(A·4)}$$

が成り立つことを示すために，

$$\sum_{u\in V_0} A^{j-1}(u,w) \leqq \alpha_{max}^{j-1} \qquad \text{(A·5)}$$

を数学的帰納法を用いて証明する．

(i) j=1 のとき，$\boldsymbol{E}$ を単位行列とすると $\sum_{u\in V_0} E(u,w) = 1$ より式 (A·5) は成り立つ．

(ii) j=k のとき，式 (A·5) は成り立つ，すなわち，

$$\sum_{u\in V_0} A^{k-1}(u,w) \leqq \alpha_{max}^{k-1} \qquad \text{(A·6)}$$

が成り立つと仮定する．式 (A·6) の両辺に α_{max} をかけると，

$$\alpha_{max} \sum_{u\in V_0} A^{k-1}(u,w) \leqq \alpha_{max}^{k} \qquad \text{(A·7)}$$

が成り立つ．このとき，

$$\alpha_{max} \sum_{u\in V_0} A^{k-1}(u,w) = \alpha_{max} A^{k-1}(0,w) + \ldots + \alpha_{max} A^{k-1}(N-1,w) \qquad \text{(A·8)}$$

である．一方で，$\boldsymbol{A}^k = \boldsymbol{A}\boldsymbol{A}^{k-1}$ より，行列 $\boldsymbol{A}^k$ の (u,w) 要素は次のように表される．$A^k(u,w) = \sum_{v=0}^{N-1} A(u,v)A^{k-1}(v,w) = A(u,0)A^{k-1}(0,w) + A(u,1)A^{k-1}(1,w) + \cdots + A(u,N-1)A^{k-1}(N-1,w)$ ここで，u について総和をとると，

$$\begin{aligned}
\sum_{u=0}^{N-1} A^k(u,w) &= A(0,0)A^{k-1}(0,w) + \ldots + A(0,N-1)A^{k-1}(N-1,w) \\
&\quad + A(1,0)A^{k-1}(0,w) + \cdots + A(1,N-1)A^{k-1}(N-1,w) \\
&\quad + \ldots \\
&\quad + A(N-1,0)A^{k-1}(0,w) + \ldots + A(N-1,N-1)A^{k-1}(N-1,w) \\
&= \{A(0,0) + \cdots + A(N-1,0)\}A^{k-1}(0,w) \\
&\quad + \{A(0,1) + \cdots + A(N-1,1)\}A^{k-1}(1,w) \\
&\quad + \ldots \\
&\quad + \{A(0,N-1) + \cdots + A(N-1,N-1)\}A^{k-1}(N-1,w)
\end{aligned}$$

が成り立つ．このとき，式 (A·8) と

$$\begin{aligned}
A(0,0) + \cdots + A(N-1,0) &= \sum_{u=0}^{N-1} A(u,0) \leqq \alpha_{max} \\
A(0,1) + \cdots + A(N-1,1) &= \sum_{u=0}^{N-1} A(u,1) \leqq \alpha_{max} \\
&\vdots \\
A(0,N-1) + \cdots + A(N-1,N-1) &= \sum_{u=0}^{N-1} A(u,N-1) \leqq \alpha_{max}
\end{aligned}$$

より，

$$\begin{aligned}
\sum_{u=0}^{N-1} A^k(u,w) &\leqq \alpha_{max} A^{k-1}(0,w) + \cdots + \alpha_{max} A^{k-1}(N-1,w) \\
&= \alpha_{max} \sum_{u\in V_0} A^{k-1}(u,w) \qquad \text{(A·9)}
\end{aligned}$$

が成り立つ．式 (A·7)，(A·9) より，

$$\sum_{u\in V_0} A^k(u,w) \leqq \alpha_{max}^k$$

が成り立つ．したがって，$j = k+1$ のときも式 (A·5) が成り立つ．

(i)，(ii) より，全ての自然数 j に対して式 (A·5) が成り立つ．以上で数学的帰納法による式 (A·5) の証明を終了する．

ここで，$H_1(v) \subseteqq V_0$ より

$$\sum_{u\in H_1(v)} A^{j-1}(u,w) \leqq \sum_{u\in V_0} A^{j-1}(u,w)$$

となるため，式 (A·4) が成り立つ．

したがって，式 (A·3) より $p_{i+j}(v) - p_{i+j-1}(v) \leqq \overline{A}(v) \sum_{w\in R_i} \Delta_i(w) \alpha_{max}^{j-1}$ となる．ここで，$R_i \subseteqq V_0$ より $\sum_{w\in R_i} \Delta_i(w) \leqq \sum_{w\in V_0} \Delta_i(w)$ が成り立ち，

$$p_{i+j}(v) - p_{i+j-1}(v) \leqq \alpha_{max}^{j-1} \Delta_i \overline{A}(v) \qquad \text{(A·10)}$$

となるため，

$$\begin{aligned}
p_{i+j}(v) &\leqq p_{i+j-1}(v) + \alpha_{max}^{j-1} \Delta_i \overline{A}(v) \leqq \ldots \\
&\leqq p_i(v) + (1 + \cdots + \alpha_{max}^{j-1}) \Delta_i \overline{A}(v) \leqq p_i(v) + j\Delta_i \overline{A}(v) \qquad \text{(A·11)}
\end{aligned}$$

が成り立つ．したがって，式 (A·2) は成り立つ．以上より，

$$(1-d) \sum_{j=1}^{\infty} d^{i+j} p_{i+j}(v) \leqq (1-d) \sum_{j=1}^{\infty} \{p_i(v) d^{i+j} + j d^{i+j} \Delta_i \overline{A}(v)\}$$

ここで，$\sum_{j=1}^{\infty} d^{i+j} \leqq \frac{d^{i+1}}{1-d}$，$\sum_{j=1}^{\infty} j d^{i+j} < \frac{d^{i+1}}{(1-d)^2}$ から

$$(1-d) \sum_{j=1}^{\infty} d^{i+j} p_{i+j}(v) < d^{i+1} p_i(v) + \frac{d^{i+1}}{(1-d)} \Delta_i \overline{A}(v)$$

が成り立つ．したがって，式 (A·1) より，

$$\begin{aligned}
r(v) &= (1-d) \sum_{j=0}^{i} d^j p_j(v) + (1-d) \sum_{j=1}^{\infty} d^{i+j} p_{i+j}(v) \\
&< (1-d) \sum_{j=0}^{i} d^j p_j(v) + d^{i+1} p_i(v) + \frac{d^{i+1}}{(1-d)} \Delta_i \overline{A}(v)
\end{aligned}$$

となるため，補題 3.2 は成り立つ． □