情報処理学会論文誌　データベース　Vol.10　No.4　11–15 (Dec. 2017)

テクニカルノート

大規模グラフに対するObjectRankの高速な近似Top-k検索

佐藤 朋紀[1,a)]　塩川 浩昭[2,b)]　山口 祐人[3,c)]　北川 博之[2,d)]

受付日 2017年6月9日, 採録日 2017年7月31日

概要：データベースに対するキーワード検索を実現する手法にObjectRankがある．ObjectRankは，データベース内のオブジェクトをグラフによって表現することでPageRankを拡張したリンク解析手法を適用し，キーワードに対する各オブジェクトの重要度を評価する．しかし，ObjectRankは重要度評価時に行列とベクトルの積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top-k検索の近似解を高速に計算する手法を提案する．提案手法は，重要度の計算過程でtop-k検索の解に影響を与えにくいノードを特定し枝刈りすることで，計算対象のノード数を削減する．本稿では，実データを用いた評価実験により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，グラフデータベース

Fast Approximate Top-k Search of ObjectRank for Large Graphs

Tomoki Sato[1,a)]　Hiroaki Shiokawa[2,b)]　Yuto Yamaguchi[3,c)]　Hiroyuki Kitagawa[2,d)]

Received: June 9, 2017, Accepted: July 31, 2017

Abstract: ObjectRank performs keyword search on databases. ObjectRank represents objects as a graph and evaluates the importance of each node with respect to the keyword based on link analysis method which extends PageRank. However, it is infeasible for ObjectRank to evaluate large graphs in a practical time since it needs to iteratively calculate the product of matrix and vector during importance evaluation. In order to address this problem, we propose a fast algorithm that efficiently approximates top-k search results obtained by ObjectRank. Our proposed algorithm reduces the number of nodes to be calculated by identifying and prunes nodes that have low influence on the top-k search results during the computation. In this paper, we evaluate the performance of our proposed algorithm by experiments using real-world data.

Keywords: ObjectRank, graph mining, graph database

1. はじめに

ObjectRank [1], [3] は，PageRank [2] 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と見なすことでPageRankと同様のリンク解析手法を適用し，キーワードに対する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．

ObjectRankは，クエリが与えられると行列ベクトル積の繰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，ノード数が百万を超えるような大規模なグラフを対象としたとき現実的な時間で

[1] 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Engineering, Tsukuba, Ibaraki 305–8573, Japan
[2] 筑波大学計算科学研究センター
Center for Computational Sciences, Tsukuba, Ibaraki 305–8577, Japan
[3] Indeed Japan, Chiyoda, Tokyo 100–0006, Japan
[a)] t.sato@kde.cs.tsukuba.ac.jp
[b)] shiokawa@cs.tsukuba.ac.jp
[c)] yyamaguchi@indeed.com
[d)] kitagawa@cs.tsukuba.ac.jp

クエリ応答を行うことが難しくなる．

本研究では，ObjectRank の Top-k 検索の近似解を高速に計算する手法を提案する．提案手法は重要度評価の計算過程において重要度が著しく低くなるノードを推定し逐次的にグラフから枝刈りする．提案手法はこの枝刈り手法により計算対象となるノード数を削減し ObjectRank の高速化を図る．

本研究の貢献は下記のとおりである．

- **高速性**：提案手法は ObjectRank と比較して 7 倍程度高速に重要度評価を行うことができる．提案手法は 100 万ノード，500 万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（4.1 節）．
- **近似精度**：提案手法は高精度に ObjectRank の Top-k 検索結果を近似することができる．本研究では実データを用いた実験を通じて提案手法が上位 100 件の結果を 90%以上の精度で求めることができることを示した（4.2 節）．

上述のとおり，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．提案手法により，様々なアプリケーションで現れるグラフデータを現実的な時間で解析することが可能となる．

2. 前提知識

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，およ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）．各ノードにラベルが付与され，ラベルによって種類が区別される．各エッジには重みが与えられ，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，1 つのノードから出るエッジの重みの総和は 1 以下になるように設定する．**図 1** に文献データベースを表現した *Schema Graph* の例を示す．図 1 の例では，“Conference”，“Author” などの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，出次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 2** に図 1 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．また，*Data Graph* の各ノードは “VLDB” や “2004” などのキーワードを保持する．

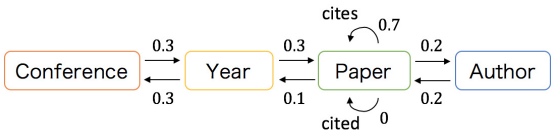

図 1　文献データベースの *Schema Graph*
Fig. 1　A *Schema Graph* of bibliographic database.

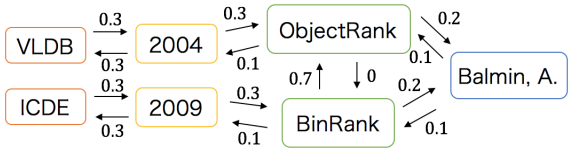

図 2　図 1 に基づいて作成された *Data Graph* の例
Fig. 2　A *Data Graph* constructed based on Fig. 1.

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．あるキーワード t に対する各ノードの ObjectRank スコアを並べたベクトル $\boldsymbol{r}_t = [r_t(v_1), \ldots, r_t(v_N)]^T$ は以下の式で得られる．

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

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，$\boldsymbol{q}_t$ は N 次元のクエリベクトルであり，$v \in BS(t)$ のとき $q_t(v) = 1/|BS(t)|$，$v \notin BS(t)$ のときは $q_t(v) = 0$ となる．

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．

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

$\boldsymbol{r}_t^{(i)}$ は i 回目の繰返し計算によって求めたスコアベクトルである．$\| \boldsymbol{r}_t^{(i+1)} - \boldsymbol{r}_t^{(i)} \|$ が十分小さくなったとき，スコアベクトルが収束したとして繰返し計算を終了する．

3. 提案手法

3.1 手法の概要

提案手法は，計算過程で ObjectRank スコアの収束値が閾値 ϵ を下回るノードを特定し，それらを逐次的にグラフから枝刈りすることで ObjectRank スコアの計算対象のノード数を削減する．枝刈り対象となるノードを効率的に発見するために，我々は理論的に ObjectRank スコア $\boldsymbol{r}_t$ の上限値 $\overline{\boldsymbol{r}}_t$ と下限値 $\underline{\boldsymbol{r}}_t$ を導出した．提案手法は下限値 $\underline{\boldsymbol{r}}_t$ を求めることで $O(1)$ で上限値 $\overline{\boldsymbol{r}}_t$ を計算する．ここで求めた上限値 $\overline{\boldsymbol{r}}_t$ は $\overline{\boldsymbol{r}}_t > \boldsymbol{r}_t$ の性質を満たすため，上限値が閾値 ϵ を下回ったとき枝刈り可能となる．

3.2　上限値と下限値の導出

$\boldsymbol{p}_t^{(i)}$ を長さ i のランダムウォークの確率を表す N 次元ベクトルとする．$\boldsymbol{p}_t^{(i)} = \boldsymbol{A}^i \boldsymbol{q}_t$ で計算し，$i = 0$ のとき $\boldsymbol{p}_i = \boldsymbol{q}$ である．このとき，ObjectRank スコアの下限値 $\underline{\boldsymbol{r}}_t$ と上限値 $\overline{\boldsymbol{r}}_t$ を以下に定義する．

定義 3.1（下限値）．i 番目の繰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．

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

定義 3.2（上限値）．i 番目の繰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．

$$\overline{r}_t^{(i)}(v) = (1-d)\sum_{j=0}^{i} d^j p_t^{(j)}(v) + d^{i+1} p_t^{(i)}(v) + \frac{d^{i+1}}{(1-d)}\Delta_t^{(i)}\overline{A}(v) \tag{4}$$

また，$\Delta_t^{(i)}$ は次のように計算する．

$$\Delta_t^{(i)} = \begin{cases} 1 & (i=1) \\ \sum_{u \in V_0} \Delta_t^{(i)}(u) & (otherwise) \end{cases}$$

ただし，G_i を i 番目の繰返し計算における部分グラフ，G_0 を元のグラフとし，V_0 は G_0 のノード集合を表す．$\Delta_t^{(i)}(v)$ は $\Delta_t^{(i)}(v) = max\{p_t^{(i)}(v) - p_t^{(i-1)}(v), 0\}$ により計算する．$\overline{\boldsymbol{A}}$ 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 番目の繰返し計算において，下限値 $\underline{r}_t^{(i)}(v)$ は次の性質を満たす．

$$\underline{r}_t^{(i)}(v) \leq r_t(v) \tag{5}$$

証明． 式 (2) より，

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

最終的な ObjectRank のスコアベクトルは $\boldsymbol{r}_t^{(i)}$ の収束値であるため，$\boldsymbol{r}_t = \boldsymbol{r}_t^{\infty}$ 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，

$$\boldsymbol{r}_t = d^{\infty}\boldsymbol{A}^{\infty}\boldsymbol{r}_t^{(0)} + (1-d)\sum_{j=0}^{\infty} d^j \boldsymbol{p}_t^{(j)} = (1-d)\sum_{j=0}^{\infty} d^j \boldsymbol{p}_t^{(j)} \tag{6}$$

が成り立つ．したがって，

$$\boldsymbol{r}_t = (1-d)\sum_{j=0}^{\infty} d^j \boldsymbol{p}_t^{(j)} \geq (1-d)\sum_{j=0}^{i} d^j \boldsymbol{p}_t^{(j)} \tag{7}$$

が成り立つため，補題 3.1 の式 (5) が成立する．　□

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

$$\overline{r}_t^{(i)}(v) > r_t(v) \tag{8}$$

証明． スペースの都合上により本稿では省略する．詳細は文献 [4] の補題 3.2 を参照されたい．　□

重要度評価の際の繰返し計算において，下限値 $\underline{\boldsymbol{r}}_t$ と上限値 $\overline{\boldsymbol{r}}_t$ 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）．i 番目の繰返し計算において，下限値 $\underline{r}_t(v)$ と上限値 $\overline{r}_t(v)$ は次式で計算する．また，次式は $O(1)$ で計算可能である．

$$\underline{r}_t^{(i)}(v) = \begin{cases} (1-d)q_t(v) & (i=0) \\ \underline{r}_t^{(i-1)} + (1-d)d^i p_t^{(i)}(v) & (i \neq 0) \end{cases} \tag{9}$$

$$\overline{r}_t^{(i)}(v) = \begin{cases} q_t(v) + \frac{d}{(1-d)}\overline{A}(v) & (i=0) \\ \underline{r}_t^{(i-1)} + d^i p_t^{(i)}(v) + \frac{d^{i+1}}{(1-d)}\Delta_t^{(i)}\overline{A}(v) & (i \neq 0) \end{cases} \tag{10}$$

証明． $i = 0$ のとき，定義 3.1 より

$$\begin{aligned} \underline{r}_t^{(i)}(v) &= (1-d)\sum_{j=0}^{i} d^j p_t^{(j)}(v) \\ &= (1-d)p_t^{(0)}(v) = (1-d)q_t(v) \end{aligned}$$

が成り立つ．また，定義 3.2 より

$$\begin{aligned} \overline{r}_t^{(i)}(v) &= (1-d)q_t(v) + dq_t(v) + \frac{d}{(1-d)}\overline{A}(v) \\ &= q_t(v) + \frac{d}{(1-d)}\overline{A}(v) \end{aligned}$$

が成り立つ．このとき，d，$q_t(v)$，$\overline{A}(v)$ は定数であるため，式 (9)，(10) は $O(1)$ で計算可能である．

次に，$i \neq 0$ のとき，定義 3.1 より $\underline{r}_t^{(i)}(v) - \underline{r}_t^{(i-1)}(v) = (1-d)d^i p_t^{(i)}(v)$ となり，

$$\underline{r}_t^{(i)}(v) = \underline{r}_t^{(i-1)} + (1-d)d^i p_t^{(i)}(v)$$

が成り立つ．また，定義 3.1，3.2 より $\overline{r}_t^{(i)}(v) - \underline{r}_t^{(i-1)}(v) = d^i p_t^{(i)}(v) + \frac{d^{i+1}}{(1-d)}\Delta_t^{(i)}\overline{A}(v)$ となり，

$$\overline{r}_t^{(i)}(v) = \underline{r}_t^{(i-1)}(v) + d^i p_t^{(i)}(v) + \frac{d^{i+1}}{(1-d)}\Delta_t^{(i)}\overline{A}(v)$$

が成り立つ．$\underline{r}_t^{(i-1)}(v)$ は $i-1$ 番目の繰返し計算においてあらかじめ計算されており，d，$p_t^{(i)}(v)$，$\overline{A}(v)$ は定数であるため，式 (9)，(10) は $O(1)$ で計算できる．　□

Algorithm 1 Our proposed algorithm

Input: G, given graph; t, keyword; ϵ, threshold; τ, maximum number of iterations
Output: Top-k nodes from final subgraph G_i
1: $i \leftarrow 0$
2: $G_i \leftarrow G$
3: **while** $i < \tau$ **do**
4: **for each** $v \in G_i$ **do**
5: calculate $r_t^{(i)}(v)$ by equation (2)
6: calculate $\underline{r}_t^{(i)}(v)$ and $\overline{r}_t^{(i)}(v)$ by equation (9) and (10)
7: **end for**
8: **if** all of the importance of $v \in G_i$ converges **then**
9: stop algorithm
10: **end if**
11: construct G_{i+1} by pruning nodes whose upper bound $\overline{r}_t^{(i)}$ is smaller than ϵ from G_i
12: $i \leftarrow i + 1$
13: **end while**

3.3　アルゴリズム

提案手法のアルゴリズムを Algorithm 1 に示す．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，最終的な部分グラフ G_i から ObjectRank スコアが上位 k 件のノードを出力する．i（$i = 0, 1, \ldots$）番目の繰返し計算において，各ノードの ObjectRank スコア $r_t^{(i)}(v)$，上限値 $\overline{r}_t^{(i)}(v)$，下限値 $\underline{r}_t^{(i)}(v)$ を計算する（5，6 行目）．このとき，補題 3.3 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（11 行目）．i 番目の繰返し計算において，G_i に含まれるすべてのノードの ObjectRank スコアが収束したとき，または繰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．

4.　評価実験

本章では，実データに対して提案手法および ObjectRank を実行し，実行速度と検索結果上位 k 件の近似精度の観点で提案手法の有効性を検証する．データセットは AMiner [*1] が公開している文献データベースを用いた．ノード数とエッジ数はそれぞれ 1,238,266，5,149,294 である．ダンピングファクタ d は文献 [1] と同様に $d = 0.85$ とした．プログラムは C++ で実装し，計算機は CPU が Intel Xeon 3.5 GHz，メモリが 128 GB の Linux サーバを用いた．

4.1　実行時間

閾値 ϵ を 0 から 1 まで変化させたときの ObjectRank と提案手法の実行時間をそれぞれ評価した．単一キーワードについて実行した結果を**図 3** に示す．実験結果より，閾値の値が小さい場合は ObjectRank の方が結果を高速に得られることが分かった．閾値の値が小さいときは枝刈りが行われにくく，上限値と下限値の計算によるオーバヘッドが影響したと考えられる．一方，閾値の値が大きい場合は枝刈りがうまく行われ，提案手法の方が高速となる．

[*1] http://aminer.org

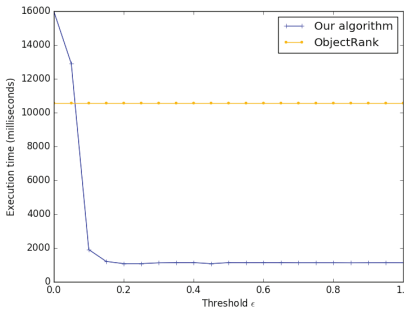

図 3　実行時間（ms）
Fig. 3　Execution time [ms].

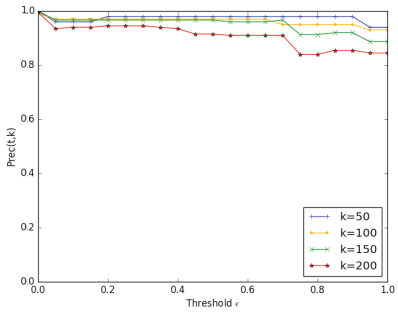

図 4　近似精度
Fig. 4　Precision.

4.2　近似精度

ObjectRank によって得られた結果を真値とし，様々な閾値 ϵ の値に対する提案手法の近似精度を計測した．近似精度の指標には式 (11) を用いた．

$$Prec(t,k) = \frac{|ProSet(t,k) \cap ORSet(t,k)|}{k} \qquad (11)$$

$ProSet(t,k)$ 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．

単一キーワードについて実行した結果を**図 4** に示す．実験結果より，上位 50 件程度であれば 90%以上の高い精度で近似解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 ObjectRank スコアの上限値と下限値を計算することで ObjectRank スコアの収束値が閾値 ϵ を下回るノードを枝刈りしたが，実験結果より枝刈りした場合に近似精度に与える影響の少ないノードが適切に選択されたことが分かる．

5.　おわりに

本稿では ObjectRank の Top-k 検索の近似解を高速に計算する手法を提案した．提案手法は ObjectRank スコアの上限値と下限値を推定し，逐次的にノードを枝刈りすることで計算対象のノード数を削減する．実データを用いた評価実験により，提案手法は 90%以上の精度を維持したまま約 7 倍程度高速に計算できることを示した．

謝辞　本研究は JSPS 科研費 JP16H06650 の助成を受けたものである．

参考文献

[1] Balmin, A., Hristidis, V. and Papakonstantinou, Y.: ObjectRank: Authority-Based Keyword Search in Databases, *Proc. VLDB*, pp.564–575 (2004).
[2] Brin, S. and Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Proc. WWW*, pp.107–117 (1998).
[3] Hristidis, V., Hwang, H. and Papakonstantinou, Y.: Authority-Based Keyword Search in Databases, *ACM Trans. Database Syst.*, Vol.33, No.1 (2008).
[4] 佐藤朋紀，塩川浩昭，山口祐人，北川博之：大規模グラフに対する ObjectRank の高速化，第 9 回データ工学と情報マネジメントに関するフォーラム (2017).

佐藤 朋紀　（正会員）

2017 年筑波大学情報学群情報科学類卒業．2017 年同大学大学院システム情報工学研究科博士前期課程在学中．大規模グラフ分析技術の高速化の研究に従事．日本データベース学会学生会員．

塩川 浩昭　（正会員）

2009 年筑波大学第三学群情報学類卒業．2011 年同大学大学院システム情報工学研究科博士前期課程修了．2011 年 4 月から日本電信電話株式会社勤務の後，2015 年筑波大学大学院博士後期課程修了，博士（工学）．2015 年より筑波大学計算科学研究センター助教．データベース，データマイニング，特に大規模グラフ分析アルゴリズムの高速化に関する研究に従事．日本データベース学会，ACM，AAAI 各会員．

山口 祐人

2014 年筑波大学大学院システム情報工学研究科博士後期課程修了．博士（工学）．米カーネギーメロン大学客員研究員，筑波大学博士研究員，産業技術総合研究所テニュアトラック研究員を経て，現在，Indeed Japan ソフトウェアエンジニア．機械学習，データマイニング等の研究に従事．日本データベース学会会員．

北川 博之　（正会員）

1978 年東京大学理学部物理学科卒業．1980 年同大学大学院理学系研究科修士課程修了．日本電気（株）勤務の後，筑波大学電子・情報工学系講師，同助教授を経て，現在，筑波大学計算科学研究センター教授．理学博士（東京大学）．データベース，情報統合，データマイニング，情報検索等の研究に従事．日本データベース学会前会長，電子情報通信学会フェロー，ACM，IEEE，日本ソフトウェア科学会各会員．本会フェロー．

（担当編集委員　青野 雅樹）