2E5-OS-09b-3

ZDDを用いたPersonalized PageRankの高速計算法

Fast Computation of Personalized PageRank with ZDDs

西野 正彬[*1] 安田 宜仁[*1] 湊 真一[*2] 永田 昌明[*1]
Masaaki Nishino Norihito Yasuda Shin-ichi Minato Masaaki Nagata

[*1]日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

[*2]北海道大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University

We propose an algorithm for accelerating the computation of Personalized PageRank (PPR). We utilize the fact that a large part of the computational cost of PPR is incurred by matrix multiplications with an adjacency matrix. To reduce the number of scalar additions and multiplications needed for matrix multiplications, we propose a method based on Zero-suppressed Binary Decision Diagram (ZDD), a data structure used for succinctly representing a family of sets, to represent the adjacency matrix in a compressed form. With regard to multiplications between matrices, our ZDD-based representation enables the sharing of the equivalent partial solutions of a multiplication, making the total number of operations proportional to ZDD size. Our approach can compute PPR more than 300 % faster than is possible using standard sparse matrix representations.

1. はじめに

PageRank [Page 99] は，Web ページに限らず，有向グラフが与えられたときにノードの重要度を計算する技術として広く用いられている．また，PageRank の拡張であり，適応的に重要度を計算可能な Personalized PageRank (PPR) [Haveliwala 02, Jeh 03] も同様に広く利用されている．PageRank および PPR を計算するための方法としては，行列とベクトル（行列）の乗算を繰り返すべき乗法に基づく方法が最も一般的である．しかし，べき乗法にはグラフが小規模な場合は問題ないが，グラフが大規模になると一度の乗算における計算回数が膨大になるため，高速な計算方法が必要となる．

本稿ではべき乗法の計算における行列間の乗算を高速化することによって PPR を高速に計算する方法を示す．具体的には，隣接行列をゼロサプレス型二分決定グラフ *(Zero-suppressed Binary Decision Diagrams: ZDD)* を用いて圧縮した形式で表現することによって，行列間の乗算に必要な実数の加算・乗算の回数を削減することによって計算を高速化する．実験では複数のデータセットに対して提案手法を適用し，従来な疎行列表現に基づく行列の乗算を用いたときと比較して最大でおよそ 300% PPR の計算を高速化できることを確認した．

2. ZDD

ZDD [Minato 93] は集合族の表現に特化した二分決定グラフ (Binary Decision Diagrams: BDD) [Bryant 86] の変種である．ZDD は集合族を無閉路有向グラフ (Directed Acyclic Graph: DAG) として表現する．図 1 は，シンボル a, b, c から構成された集合族 $\{\{a,b,c\},\{a,c\},\{b,c\}\}$ を表した ZDD である．ZDD のノードは 2 種類あり，図中の丸いノードを中間ノード，四角いノードを終端ノードとよぶ．すべての中間ノードは対応するシンボルをラベルとしてもち，必ず 0-枝 (図中破線) と 1- 枝 (図中実線) とよばれるふたつのリンクをもつ．リンクをもたないノードは終端ノードとよばれ，いずれの ZDD も必ず

連絡先: nishino.masaaki@lab.ntt.co.jp

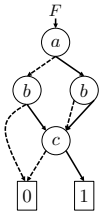
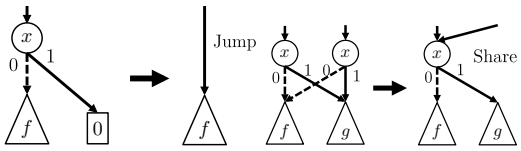


図 1: 集合族 $\{\{a,b,c\},\{a,c\},\{b,c\}\}$ を表す ZDD の例

0-終端ノードと 1-終端ノードの 2 つを持つ．ZDD の先頭にあるノードは根とよばれ，根から 1-終端ノードまでの 1 つのパスが集合族に含まれる 1 つの集合に対応する．例えば図 1 の ZDD では根から 1-終端ノードに至るパスは $a \longrightarrow b \longrightarrow c \longrightarrow 1$, $a \longrightarrow b \dashrightarrow c \longrightarrow 1$, そして $a \dashrightarrow b \longrightarrow c \longrightarrow 1$ の 3 種類であり，それぞれが集合 $\{a,b,c\}$, $\{a,c\}$, $\{b,c\}$ に対応している．ZDD の全てのパスにおいてシンボルの出現順序が一定であるとき，ZDD は順序づけされているという．図 1 の ZDD では $a < b < c$ の順序で順序づけされている．

ZDD によって集合族を表現することの利点として，巨大な集合族を圧縮された小さなグラフとして表現できることが挙げられる．同じ集合族を表現する等価な順序づけされた ZDD は複数種類存在するが，図 2 に示す 2 種類の簡約化規則を可能な限り適用することによって最もノード数が少ない既約な ZDD を得ることができる．

ZDD の計算機上での表現について述べる．ZDD は，各ノードごとに対応するシンボル，1-枝が指すノードのアドレス，0-枝が指すノードのアドレスの 3 つのフィールドを用意することで表現することができる．ある ZDD f の総ノード数を $B(f)$ とすると，ZDD は上記 3 フィールドを格納する要素数 $B(f)$ の配列で表現できる．配列中の k 番目のノードに対応するシンボルを $v(k)$, 1-枝，0-枝が指すノードの配列中での添字を，それぞれ $h(k), l(k)$ とする．以下では ZDD のノードは大きさ $B(f)$ の配列にトポロジカルソートの逆順で格納されているものとする．すなわち，配列の 1, 2 番目の要素がそれぞれ 0, 1



(a) Node deletion. (b) Node sharing.

図 2: ZDD の簡約化規則. (a) 削除規則と (b) 共有規則

終端ノード，B(f) 番目の要素が根とする．

3. Personalized PageRank

Personalized PageRank を説明する前に，まず通常の PageRank について簡単に説明する．N 個のノードからなる有向グラフを $G = (V, E)$ とする．V はノードの集合，E はエッジの集合とする．この有向グラフを表現する隣接行列を $\mathbf{A}$ とする．A_{ij} を $\mathbf{A}$ の i 行 j 列の要素とすると，グラフ G が i 番目のノードから j 番目のノードへのエッジを含むならば $A_{ij} = 1/O_i$，そうでないならば $A_{ij} = 0$ とする．ここで O_i は i 番目のノードの出次数である．次に，$\mathbf{x}$ をノードの集合 V 上での確率分布を表す N 要素の非負実数ベクトルとする．すなわち，すべての $1 \leq i \leq N$ について $x_i \in [0, 1]$ かつ $\sum_{i=1}^{N} x_i = 1$ を満たすとする．グラフ G の PageRank ベクトルとは，$\mathbf{x}$ のうち，

$$\mathbf{x} = (1 - c)\mathbf{A}\mathbf{x} + c\mathbf{u} \tag{1}$$

を満たすもののことである*1．ここで $\mathbf{u}$ は嗜好ベクトルとよばれる N 次元実数ベクトルであり，$\mathbf{x}$ と同様にすべての $1 \leq i \leq N$ について $u_i \in [0, 1]$ かつ $\sum_{i=1}^{N} u_i = 1$ を満たす．嗜好ベクトルは，グラフ中の各ノードに対するデフォルトの重要度を与えるためのものであり，PPR において重要な役割を占める．c はテレポーテーション定数とよばれる実数パラメータであり，$c \in [0, 1]$ である．

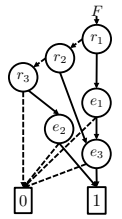
式 (1) を満たす $\mathbf{x}$ を求める方法としてはべき乗法が代表的である．すなわち，適当な初期値 $\mathbf{x}^{(0)}$ から始まり，$\mathbf{x}^{(t)} \leftarrow (1 - c)\mathbf{A}\mathbf{x}^{(t-1)} + c\mathbf{u}$ として，$\mathbf{x}^{(t)}$ を更新していく方法である．計算を繰り返すことによって $\mathbf{x}^{(t)}$ は (1) を満たす値に収束することが知られている．

3.1 Personalized PageRank

通常の PageRank の計算においては，嗜好ベクトルは $u_i = 1/N$ として，どのノードに対しても一定の重みを与えるのが一般的である．一方で嗜好ベクトルの重みを変化させることによって PageRank の値を調整することもできる．すなわち重要なノードの嗜好ベクトルの重みを大きくする，あるいは影響を無視したいノードの重みを小さくすることによって，所望の PageRank ベクトルを得ることができる．このように嗜好ベクトルを調整して計算された PageRank を Personalized PageRank (PPR) とよぶ．PPR は有用な技術であるが，異なる嗜好ベクトルが与えられるたびに毎回 PPR を計算するのは，大規模なグラフを扱う場合には現実的ではない．そこで重要となるのは [Jeh 03] で示された PPR の線形性である．PPR の線形性とは，ある 2 つの嗜好ベクトル $\mathbf{u}_1$，$\mathbf{u}_2$ と，それぞれに

*1 簡単のため出次数が $O_i = 0$ であるようなノード v_i はグラフ中には含まれないものとする．

$$\begin{pmatrix} 1.0 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.5 \\ 0.0 & 1.0 & 0.0 \end{pmatrix}$$



(a) Adjacent matrix (b) Corresponding ZDD

図 3: 隣接行列と対応する ZDD の例

対応する PageRank ベクトル $\mathbf{x}_1$，$\mathbf{x}_2$ があったときに，関係

$$\alpha\mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2 = (1 - c)\mathbf{A}(\alpha\mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2) + c(\alpha\mathbf{u}_1 + (1 - \alpha)\mathbf{u}_2)$$

が成り立つことをいう．ここで α は $\alpha \in [0, 1]$ を満たす任意の実数である．PPR の線形性を用いることで，嗜好ベクトル $\mathbf{u}_3 = \alpha\mathbf{u}_1 + (1 - \alpha)\mathbf{u}_2$ に対応する PageRank ベクトル $\mathbf{x}_3$ を，$\mathbf{x}_1$, $\mathbf{x}_2$ から $\mathbf{x}_3 = \alpha\mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2$ として求めることができる．つまり，あらかじめいくつかの嗜好ベクトルに対する PageRank ベクトルを求めておくことによって，それらの嗜好ベクトルの線形和に対しても PageRank ベクトルを求めることができる．

M 個の嗜好ベクトルを列ベクトルとして並べた $N \times M$ の行列を $\mathbf{U}$ とすると，対応する M 個の PageRank ベクトルを求める過程は，

$$\mathbf{X}^{(t)} \leftarrow (1 - c)\mathbf{A}\mathbf{X}^{(t-1)} + c\mathbf{U} \tag{2}$$

として，行列 $\mathbf{X}$ を更新するものとして表現できる．ここで $\mathbf{X}$ は $N \times M$ の行列であり，$\mathbf{X}$ の i 番目の列ベクトルが $\mathbf{U}$ の i 番目の列ベクトルに対応する PageRank ベクトルとなる．

PPR を求めるためには，(2) にあるように隣接行列 $\mathbf{A}$ と行列 $\mathbf{X}$ との積を繰り返し求める必要がある．一般に隣接行列は疎行列となるため，一度の乗算に $\mathrm{nnz}(\mathbf{A}) \times M$ 回の実数の加算と乗算が必要となる．ここで $\mathrm{nnz}(\mathbf{A})$ は $\mathbf{A}$ の非ゼロ成分の個数である．次節ではこの乗算に必要な計算を，隣接行列を ZDD を用いて表現することで削減する方法について述べる．

4. ZDD を用いた PPR の計算

ZDD を用いて効率的に行列間の乗算を行う方法について説明する．なお，本節で説明する方法は，以前に [西野 12] で示した ZDD を用いて二値行列を表現し，行列とベクトルの乗算を行う方法の拡張として位置づけられる．既存手法との違いは，PageRank および PPR の計算で用いられる列正規化行列 (列ベクトルの非ゼロ成分の値が一定である行列) を扱えるようにした点，および行列間の乗算を可能とした点にある．

まず ZDD を用いて隣接行列を表現する方法を示す．ZDD は集合族を表現するためのデータ構造であるため，隣接行列を集合族として表現することで ZDD で隣接行列を表現することができる．いま，隣接行列 $\mathbf{A}$ の集合族による表現を $\mathcal{S}_{\mathbf{A}}$ とする．$\mathbf{A}$ が $N \times N$ の行列であったとすると，その各行に対応する N 個のシンボル $r_1, \ldots, r_N$ と，各列に対応する N 個のシンボル $e_1, \ldots, e_n$，あわせて $2N$ 個のシンボルを用いて $\mathcal{S}_{\mathbf{A}}$ を

$$\mathcal{S}_{\mathbf{A}} = \bigcup_{i=1}^{N} \{\{r_i, e_{a_i(1)}, \ldots, e_{a_i(b_i)}\}\}, \tag{3}$$

Algorithm 1 ZDD で表現された隣接行列と実数行列の乗算

Input: 隣接行列を表現する ZDD $f_{\mathbf{A}}$，$N \times M$ の実数行列 $\mathbf{X}$
Output: $N \times M$ の実数行列 $\mathbf{Y} = \mathbf{AX}$

```
for k = 1, 2 do  /* 終端ノード */
    s[k] ← 0
for k = 3 to (B(f_A) − N) do  /*v(k) = e_**/
    s[k] ← s[l(k)] + s[h(k)] + 1/O_{η(v(k))} x_{η(v(k))}
for k = (B(f_A) − N) + 1 to B(f_A) do  /*v(k) = r_**/
    y_{η(v(k))} ← s[h(k)]
return Y
```

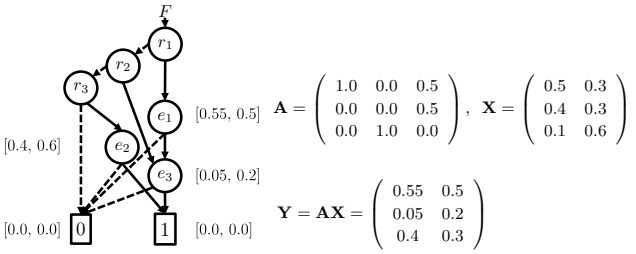


図 4: アルゴリズム 1 の実行例

として定義できる．ここで $a_i(j)$ は行列の i 行目に対応する行ベクトル $\mathbf{A}_i = (A_{i1}, \ldots, A_{iN})$ において j 個目の非ゼロ成分の位置を返す関数である．また b_i は i 行目に対応する行ベクトルの個数を表す．この定義は $\mathbf{A}$ の 1 つの行，その非ゼロ成分を要素とする集合として表現し，それらの集まりである集合族として行列を表現したものである．隣接行列の例と，そのZDD による表現を図 3 に示す．図 3(a) の隣接行列は，集合族 $\{\{r_1, e_1, e_3\}, \{r_2, e_3\}, \{r_3, e_2\}\}$ として表現される．ここで $a_1(1) = 1$, $a_1(2) = 3$, $a_2(1) = 3$, $a_3(1) = 2$, $b_1 = 2$, $b_2 = 1$, そして $b_3 = 3$ である．

次に ZDD で表現された隣接行列と実数行列の乗算を実行する手続きを説明する．乗算は ZDD の構造を利用した動的計画法として実行される．すなわち，ZDD の各ノードごとに M 個の実数を記憶する配列を用意し，その値を終端ノードから根の順に再帰的に設定していくことで，最終的に所望の計算結果を得ることができる．ZDD の i 番目のノードに対して用意された記憶領域を $\mathbf{s}[i]$ と表す．手続きをアルゴリズム 1 に示す．ここで $\eta()$ はシンボル r_i，e_i を受け取り，その添字を返す関数である．例えば $\eta(e_1) = 1$，$\eta(r_3) = 3$ となる．アルゴリズムはまず 1, 2 行目で終端ノードに対応する配列 $\mathbf{s}[0]$ を $\mathbf{0}$ で初期化したのちに，3，4 行目でシンボル e_* が付与された各中間ノードの配列をその子ノードの配列を参照して設定する．最後に 5，6 行目で i 行目に対応する行ベクトルを取り出し処理を終了する．図 4 はアルゴリズム 1 の実行例である．行列計算として計算する際に，中間ノードの値がどのように更新されていくかを示している．

ZDD を用いることによって，行列同士の乗算に必要な実数の加算および乗算の回数は，シンボル e_* が対応づけられたノードの個数に等しくなる．図 4 の例では，従来法では $\mathrm{nnz}(\mathbf{A}) = 4$ 回の計算が必要であったところが，$v(k) = e_*$ であるようなノードの個数，つまり 3 回の計算で済むことが分かる．また，証明は紙面の都合上割愛するが，該当のノード数は必ず元の隣接行列の非ゼロ要素数よりも少なくなるため，ZDD を用いることで計算の回数が増えることはないのも重要な特長である．

表 1: データセット・隣接行列を表現した ZDD のノード数

データセット	ノード数	リンク数	ZDD のノード数
Wikipedia	2,394,385	5,021,410	2,041,999
AS	22,963	48,436	14,874
Web	325,729	1,497,135	432,857

4.1 手法の拡張

前節の方法で PPR を計算できるが，効率的に計算を行うためにはいくつかの工夫が不可欠である．本節では ZDD のノード数の削減のための ZDD による表現方法の工夫と，省メモリで乗算を実行するための実装上の工夫について述べる．

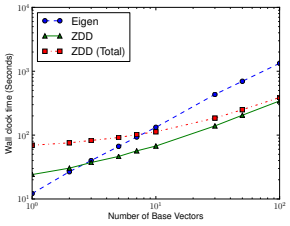
行列の分割と順序づけ　順序づけされた BDD/ZDD は，変数の順序を変化させるとノード数が変化するため，対象とする集合族に適した変数順序を用いることが重要となる．さらに，隣接行列の表現に固有の方法として，行列を列ベクトルを単位として複数の行列に分割し，それを ZDD として表現することでよりノード数を削減できることが知られている [西野 12]．以上 2 点より，適した順序づけおよび行列の分割を行うことが必要となる．BDD/ZDD において，最適な変数順序を発見する問題は NP 困難であることが知られている．そのため本稿では貪欲法に基づくヒューリスティックを用いて順序づけと行列の分割の問題を同時に解く方法を示す．まず容器を K 個用意し，次に各列ベクトルをそれらの容器のどれかに入れることを N 回繰り返す．ある列ベクトルを入れる容器を選択する際には，その列ベクトルと各容器に入っている列ベクトルの集合との類似度をもとにスコアを算出し，類似度が最も高くなった容器を選択する．同じ容器に入った列ベクトルを一つのグループとするように行列を分割することによって，分割と並べ替えを同時に実行することができる．

必要な記憶領域の削減　アルゴリズム 4 では計算の途中経過を記憶するために大きさ M の実数配列をノード数だけ用意する必要があった．そのため一般な疎行列表現を用いる場合と比べ必要な記憶領域が非常に大きくなる可能性がある．そこで，アルゴリズム 1 の実行中に，あるノードに対応付けら得た配列がそれ以降の計算で利用されるかどうかを判別し，以降の計算で利用されない記憶領域を開放するようにする．詳細は紙面の都合上割愛するが，乗算で必要となる記憶領域は常に $2N$ を超えないため，記憶領域を大幅に削減することができる．それ以降利用されるかどうかの判定を行うためには，あるノードが以降何度参照されるかを記憶する参照カウンタを保持する必要がある．参照カウンタの保持にはさらに追加で記憶領域が必要となるが，M が大きいときにはメモリの削減の効果のほうが大きくなる．

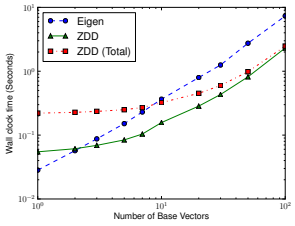
5. 検証

提案手法によって Personalized PageRank を計算し，既存手法と比較した．データセットとして表 1 左に示す 3 種類の有向グラフデータを用いた．比較対象として，疎行列と行列の乗算で広く使われるデータ構造である Compressed Row Storage (CRS) [Barrett 94] を実装した行列計算ライブラリ，Eigen[*2]のバージョン 3.1.2 を利用した．実装言語は C++，コンパイラは gcc 4.6.3 (-O3)，実行環境として 2.66GHz Intel Xeon CPU と 256GB RAM を搭載した Linux をそれぞれ用いた．

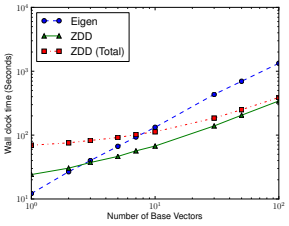
*2 http://eigen.tuxfamily.org/index.php



(a) Wikipedia



(b) AS



(c) Web

図 5: PPR の計算時間の比較

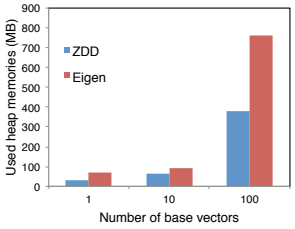


図 6: メモリ使用量

5.1 実験結果

図 5 は，嗜好ベクトルの個数を横軸，PPR の 50 回の繰り返し計算にかかった計算時間を縦軸として計算時間を比較したものである．図中の 3 種類のプロットは，それぞれ Eigen による計算時間，提案手法による計算時間，行列の並べ変えおよび ZDD の構築にかかる時間を含めた提案法による計算時間を表している．Eigen による計算結果と ZDD による PPR の計算の時間のみを比較すると，嗜好ベクトルの数が 3 個以上のときであれば常に提案手法のほうが高速に PPR を計算できていることが分かる．特に 20 個以上の嗜好ベクトルを対象とする場合には，最大で 300%の高速化に成功している．表 1 右に各隣行列を ZDD で表現したときのラベル e_* をもつノード数を表している．いずれのデータでも隣接行列の非ゼロ要素数（リンク数）と比較すると最大 1/3 程度になっていることから，計算にかかった時間もノード数の削減に併せて 1/3 程度まで削減されたものと解釈できる．一方，嗜好ベクトルのサイズが 1 または 2 のときは，既存手法のほうが高速に計算できるという結果が得られた．これは，嗜好ベクトルが小さいときには CPU キャッシュヒット率の差の影響が顕著となるためと考えられる．キャッシュヒット率については，CRS フォーマットは行列の乗算が CPU キャッシュを利用しやすい形式である [Barrett 94] のに対して，ZDD を用いた表現では子ノードの値の参照時に局所的でないメモリアクセスが発生するため，キャッシュヒット率を高めることができない．

前処理を含めた場合であっても，嗜好ベクトルのサイズが大きい場合であれば提案手法は既存手法と比較して高速に動作した．前処理にかかる時間は嗜好ベクトルの個数，および乗算の繰り返し回数によらず一定であるため，多数の嗜好ベクトルを利用する場合，また，乗算の繰り返し回数が多い場合には提案手法はより処理の高速化に寄与するといえる．

嗜好ベクトルの個数を変化させたときの利用ヒープメモリ容量を図 6 に示す．参照カウンタを用いた提案手法では使用メモリ量を既存手法よりも削減できていることが分かる．

6. 関連研究

PageRank および PPR の高速な計算方法について，これまで多くの検討がなされているが，線形代数に基づく手法とモンテカルロ法に基づく方法とに分類することができる．線形代数に基づく手法では，外挿を用いた繰り返し計算回数の削減 [Kamvar 03b] や，グラフの性質を利用して，行列を複数のブロックに分割する方法 [Kamvar 03a] などが提案されている．提案手法の利点の一つとして，これら既存手法と容易に組み合わせることが可能なことが挙げられる．すなわち，既存の線形代数に基づく手法では，内部的に隣接行列と実数行列との乗算を行うため，その部分を既存手法で置き換えることで，さらなる高速化が達成できると考える．モンテカルロ法に基づく方法は，グラフ上でランダムウォークシミュレーションを行うことによって，近似的にノード上の確率分布を求める [Fogaras 05]．線形代数にに基づく手法と比較して高速な手法が提案されているが，正しい値への収束を保証できないという課題がある．

7. おわりに

本稿では ZDD を用いて隣接行列を表現することによって，ボトルネックである行列間の乗算を高速化して Personalized PageRank の計算を高速化できることを示した．提案手法は疎行列を対象とした乗算を高速化する一般的な方法であるため，PPR によらず様々な場面での活用が期待できる．今後は他の活用事例についても検討を行っていきたい．

参考文献

[Barrett 94] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Vorst, der H. V.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*, SIAM, Philadelphia, PA (1994)

[Bryant 86] Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677–691 (1986)

[Fogaras 05] Fogaras, D., Rácz, B., Csalogány, K., and Sarlós, T.: Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments, *Intermet Mathematics*, Vol. 2, No. 3 (2005)

[Haveliwala 02] Haveliwala, T. H.: Topic-Sensitive PageRank, in Proceedings of WWW, pp. 517–526 (2002)

[Jeh 03] Jeh, G. and Widom, J.: Scaling Personalized Web Search, in Proceedings of WWW, pp. 271–279 (2003)

[Kamvar 03a] Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H.: Exploiting the Block Structure of the Web for Computing PageRank, Technical report, Stanford University (2003)

[Kamvar 03b] Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H.: Extrapolation Metdhods for Accelerating PageRank Computations, in Proceedings of WWW, pp. 261–270 (2003)

[Minato 93] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatiorial Problems, in Proc. of DAC'93, pp. 272–277 (1993)

[Page 99] Page, L., Brin, S., Motwani, R., and Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Stanford University (1999)

[西野 12] 西野 正彬, 安田 宜仁, 小林 透：ZDD を用いた効率的な集合拡張の計算, 人工知能学会論文誌, Vol. 27, No. 2, pp. 22–27 (2012)