2F3-3

ZDDを用いた効率的な集合拡張の計算

An Efficient Calculation of Set Expansion using Zero-Suppressed Binary Decision Diagrams

西野 正彬　安田 宜仁　小林 透
Masaaki Nishino　Norihito Yasuda　Toru Kobayashi

日本電信電話株式会社 NTT サイバーソリューション研究所
NTT Cyber Solutions Laboratories, NTT Corporation

Set Expansion is a method for finding similar sets of items from a seed set. It is useful on examining a large set of items to find hidden relationships among them. In this paper, we propose a method for reducing the number of calculations needed on executing *Bayesian Sets*, which is one of the most popular set expansion algorithms. The key point of our method lies in the use of Zero-suppressed binary decision diagrams (ZDD) for expressing a binary value sparse matrix in a compressed form and for directly executing calculations on it. We show a method for expressing a binary value matrix with ZDD, and also show a method for reducing the size of ZDD. In experiments we show that the proposed method can reduce the number of calculations both on synthesis data and real data.

1. はじめに

集合拡張 *(Set Expansion)* とは，あるアイテムの集合 $\mathcal{D}$ にいくつかのアイテムからなる種 (seed) を入力として与え，その種を含む $\mathcal{D}$ の潜在的な集合の他のアイテムを出力するアルゴリズムのことである．例えばアイテムを単語とした場合，種として"elephant", "horse", "bat" といった単語が与えられたならば，集合拡張アルゴリズムは他の哺乳類の名前を出力することを期待される．集合拡張を用いれば，利用者が膨大なアイテム集合から，対話的に類似のアイテムを検索することができるため，大規模データの解析手法として有用である．

これまでに様々な集合拡張アルゴリズムが検討されてきているが [Wang 07] [Ghahramani 06] [Vickrey 10] ，その基本的な動作は，入力として与えられた種をもとにアイテム集合 $\mathcal{D}$ に含まれる各アイテムに対するスコアを計算し，それに基づいてランキングすることにある．スコア計算にはアイテム集合 $\mathcal{D}$ のサイズに比例する計算量が必要となるため，$\mathcal{D}$ が巨大になると，集合拡張を行うために必要となる計算量もそれに伴い増大する．集合拡張は利用者とのインタラクションに基づいて実行されるため，巨大なアイテム集合に対する集合拡張を実行するためには，計算量の削減による計算の高速化が必要となる．

本稿では，代表的な集合拡張アルゴリズムのひとつである Bayesian Sets[Ghahramani 06] について，その計算量を削減する手法を提案する．手法のポイントは，合致度スコアを計算する際に必要となる疎行列とベクトルとの乗算の計算回数を，**ゼロサプレス型二分決定グラフ** *(ZDD: Zero-suppressed Binary Decision Diagrams)*[Minato 93] を行列の表現手法として用いることによって削減する点にある．ZDD は組み合わせ集合を圧縮した形式で表現することが可能なデータ構造である．ここで組み合わせ集合とは，n 個のアイテムから任意個のアイテムを選んだ組み合わせを要素とする集合のことである．本稿では ZDD を用いて疎な 2 値行列を表現する手法，および ZDD の構造を利用して実数ベクトルと 2 値行列との積を効率的に計算する手法を述べる．

連絡先: 西野 正彬, NTT サイバーソリューション研究所,
神奈川県横須賀市光の丘 1-1,
nishino.masaaki@lab.ntt.co.jp

2. Bayesian Sets

Bayesian Sets は Ghahramani らによって提案された集合拡張アルゴリズムであり，ベイズ推定の枠組みを用いて集合拡張を定義していることを特徴とする．$\mathcal{D}$ をアイテム集合，$\mathbf{x} \in \mathcal{D}$ をアイテム集合に含まれるアイテムとする．ユーザから与えられる種 (小さなアイテム集合) を $\mathcal{D}_c$ とする．種 $\mathcal{D}_c$ は $\mathcal{D}$ の部分集合である．Bayesian Sets では，種 $\mathcal{D}_c$ が与えられたときに，各アイテム $\mathbf{x} \in \mathcal{D}$ の $\mathcal{D}_c$ への合致度スコア score($\mathbf{x}$) を以下のように定義する．

$$\mathrm{score}(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{D}_c)}{p(\mathbf{x})} \tag{1}$$

アイテム $\mathbf{x}$ の出現確率が，パラメタ θ をもつ確率分布 $p(\mathbf{x}|\theta)$ によって定義されるとすると，(1) を構成する各確率分布は以下のように定義される．

$$p(\mathbf{x}) = \int p(\mathbf{x}|\theta)p(\theta)d\theta \tag{2}$$

$$p(\mathcal{D}_c) = \int \prod_{\mathbf{x}_i \in \mathcal{D}_c} p(\mathbf{x}_i|\boldsymbol{\theta})p(\theta)d\theta \tag{3}$$

$$p(\mathbf{x}|\mathcal{D}_c) = \int p(\mathbf{x}|\theta)p(\theta|\mathcal{D}_c)d\theta \tag{4}$$

$$p(\theta|\mathcal{D}_c) = \frac{p(\mathcal{D}_c|\theta)p(\theta)}{p(\mathcal{D}_c)} \tag{5}$$

以下では $\mathbf{x}_i \in \mathcal{D}_c$ が M 次元の 2 値ベクトル $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iM})$ であり，かつ．Bernoulli 分布に従うとする．ただし $x_{ij} = \{0, 1\}$ である．$p(\mathbf{x}_i|\boldsymbol{\theta})$ は

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \prod_{j=1}^{M} \theta_j^{x_{ij}}(1-\theta_j)^{1-x_{ij}} \tag{6}$$

となる．Bernoulli 分布の共役事前分布は Beta 分布であるから，$\boldsymbol{\theta}$ を

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{j=1}^{M} \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \theta_j^{\alpha_j - 1}(1-\theta_j)^{\beta_j - 1} \tag{7}$$

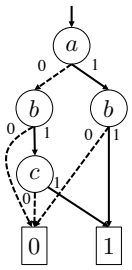


図 1: 組み合わせ集合 $\{ab, bc\}$ を表す ZDD の例

という確率分布に従うものとする．ここで $\boldsymbol{\alpha}$ と $\boldsymbol{\beta}$ は Beta 分布のハイパーパラメタである．

上記の分布を (1) に当てはめると，score($\mathbf{x}$) は

$$\mathrm{score}(\mathbf{x}) = \prod_{j=1}^{M} \frac{\alpha_j + \beta_j}{\alpha_j + \beta_j + K} \left(\frac{\tilde{\alpha}_j}{\alpha_j}\right)^{x_{\cdot j}} \left(\frac{\tilde{\beta}_j}{\beta_j}\right)^{1-x_{\cdot j}} \tag{8}$$

となる．ここで K は D_c に含まれるアイテムの総数，$\tilde{\alpha}_j = \alpha_j + \sum_{i=1}^{K} x_{ij}$, $\tilde{\beta}_j = \beta_j + K - \sum_{i=1}^{K} x_{ij}$ とする．式 (8) の対数をとると，

$$\log \mathrm{score}(\mathbf{x}) = c + \sum_{j=1}^{M} q_j x_{\cdot j} \tag{9}$$

として，$\mathbf{x}$ の線形式として表すことができる．ここで c, q_j はそれぞれ

$$c = \sum_{j=1}^{M} \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + K) + \log\tilde{\beta}_j - \log\beta_j \tag{10}$$

$$q_j = \log\tilde{\alpha}_j - \log\alpha_j - \log\tilde{\beta}_j + \log\beta_j \tag{11}$$

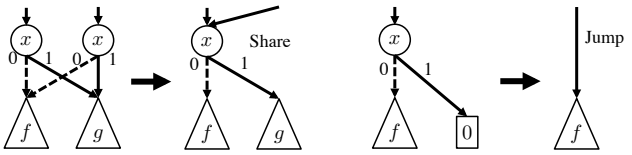
として表される．$\mathcal{D}$ を，その各行がひとつのアイテムに対応するような M 列の行列 $\mathbf{X}$ として表すとすると，各アイテムに対するスコアの計算は行列とベクトルの積として

$$\mathbf{s} = c + \mathbf{X}\mathbf{q} \tag{12}$$

として表すことができる．c は $\mathbf{x}$ の値に依存しないので，Bayesian Sets アルゴリズムは，$\mathcal{D}_c$ をもとに q_j を計算し，$\mathbf{X}$ と $\mathbf{q}$ との積を計算することで実行することができると解釈することができる．種 $\mathcal{D}_c$ は数個のアイテムを含む程度であると考えられるので，Bayesian Sets アルゴリズムを効率的に実行するためには，行列とベクトルとの積 $\mathbf{X}\mathbf{q}$ の計算量を削減する必要がある．

3. ZDD

ZDD は，Minato[Minato 93] によって提案された，組み合わせ集合の表現に特化した二分決定グラフ (BDD: Binary Decision Diagrams) [Bryant 86] である．ZDD は指向性をもったループのないグラフの構造をしており，グラフの終端以外の各節点はすべて 0-枝と 1-枝のふたつのリンクをもつ．グラフ末端の節点は終端節点とよばれ，いずれの ZDD も 0-終端節点と 1-終端節点の 2 つの終端節点を持つ．終端以外の節点は，すべて組み合わせ集合に含まれるあるシンボルに対応づけられている．シンボル a, b, c からなる組み合わせ集合 $\{ab, bc\}$ を表した ZDD の例を図 1 に示す．ある組み合わせ集合を表す ZDD の構造は，変数の順序が一定ならば同一となる．



(a) 等価なノードの共有　　(b) 冗長なノードの削除

図 2: ZDD の簡約化規則

順序づけされた ZDD に対して簡約化規則を可能な限り適用することによって，それ以上簡約化できない規約な ZDD を得ることができる．ZDD の簡約化規則には，冗長な節点の削除と等価な節点の共有の 2 種類がある．2 種類の規則による簡約化の様子を図 2 に示す．ある ZDD に対してより多くの簡約化規則を適用ことが可能ならば，最終的に得られる簡約化された ZDD の節点数を少なくすることができ，ZDD を保持するために必要なメモリ量を削減できる．

ZDD は各節点ごとに対応するシンボル，1-枝が指す節点のアドレス，0-枝が指す接点のアドレスの 3 つのフィールドを用意することで計算機上で表現することができる．ある ZDD f の総節点数を $B(f)$ 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 番目の節点に対応するシンボルを v_k, 1-枝，0-枝が指す節点の配列中での添字を，それぞれ h_k, l_k とする．なお，以下の計算を簡単にするため，配列中では $B(f) > k \geq 2$ のときに $l_k < k$, $h_k < k$, $v_{l_k} > v_k$, $v_{h_k} > v_k$ が成立しているものとする．

4. ZDD を用いた疎行列とベクトルの乗算

ZDD によって行列との積を計算する手法について述べる．まず ZDD によって 2 値行列を表現する手法を示したのちに，ZDD の構造を用いて 2 値行列と実数ベクトルとの積を計算する手法について説明する．

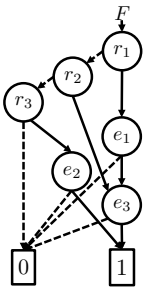
ZDD によって 2 値行列を表現する手法について述べる．まず，$\mathcal{D}$ に含まれる各アイテムが M 次元の 2 値ベクトルとし，アイテムの総数を N とすると，全てのアイテムの集合は $N \times M$ の行列 $\mathbf{X}$ として表すことができる．次に，$\mathbf{X}$ を組み合わせ集合として表現する．組み合わせ集合で用いられるシンボルの集合 S を，$S = \{r_1, \ldots, r_N, e_1, \ldots, e_M\}$ とする．ここで，$r_1, \ldots, r_N$ は，それぞれアイテムに対応するシンボルであり，$e_1, \ldots, e_M$ はアイテムを表す M 次元ベクトルの各成分にそれぞれ対応するシンボルとする．これらのシンボルを用いて，$\mathbf{X}$ に対応する組み合わせ集合 $Z_{\mathbf{X}}$ を，

$$Z_{\mathbf{X}} = \bigcup_{i=1}^{N} \{r_i e_{a_i(1)} \ldots e_{a_i(b_i)}\} \tag{13}$$

として表現する．ここで $a_i(j)$ は，アイテム集合中の i 番目のアイテム $\mathbf{x}_i$ の要素 $x_{i1}, \ldots, x_{iM}$ のうち，値が 1 となる j 番目要素の添字を表す．また b_i は $x_{i1}, \ldots, x_{iM}$ のうち値が 1 となる要素の総数である．$1 \leq a_i(1) < \cdots < a_i(b_i) \leq M$ を満たす．例えば図 3(a) の行列には，組み合わせ集合 $\{r_1e_1e_3, r_2e_3, r_3e_2\}$ が対応する．これを ZDD として表現したものが図 3(b) となる．ここで，シンボル間には $r_1 < r_2 < \cdots < r_N < e_1 < e_1 <$

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

(a) 2値行列の例



(b) 対応するZDD

図 3: 2 値行列と対応する ZDD の例

$e_2 < \cdots < e_M$ という順序を与えている．順序は変更してもよいが，任意の $1 \le i \le N, 1 \le j \le M$ について $r_i < e_j$ となるようにする．

このようにして構成された ZDD にはいくつかの特徴がある．まず，このように構成された ZDD の節点数は，$N + \sum_{i=1}^{N} b_i$ 以下であり，かつシンボル $r_1, \ldots, r_N$ に対応する節点は，常に高々1 つしか出現しないという特徴がある．次に，節点の共有による簡約化については，共有が起こりうるのは $e_1, \ldots, e_N$ に対応する節点のみとなり，かつ共有が起きる条件が，各項に含まれるシンボルのうち，順序的に後にくるものがすべて共通な場合にのみ，ZDD において構造の共有が行われるというものになる．図 3 では，行列の 1 行目に相当する項が $r_1e_1e_3$，3 行目に相当する項が r_3e_3 であり，e_3 に相当する節点で共有による圧縮が発生している．

次に，構成された ZDD を用いて行列とベクトルとの乗算を行う手続きを図 4 に示す．図のアルゴリズムでは，ZDD の構造を利用して，葉節点から順に頂点節点に向けてスコアを更新する手続きを実行する．入力として，行列 $\mathbf{X}$ の ZDD と，M 次元の実数ベクトル $\mathbf{q}$ を与える．また，記憶領域として大きさ $B(f_{\mathbf{X}})$ の実数配列 value を用意する．また，$\eta(v_k)$ はシンボルを受け取り，そのシンボルの添字を返す関数である．例えば $\eta(e_3) = 3, \eta(r_1) = 1$ である．ステップ 1-3 では value の初期化を行っている．ステップ 4-10 で，各節点について value の値を計算する．ここで節点の配列が 3. 章で導入した規則に沿って並んでいるとすると，計算は常に子孫節点から親節点の順に進むことになる．ステップ 6 では，v_k が $e_1, \ldots, e_M$ のいずれかであった場合に value を計算している．ある節点 k に対応する value(k) には，ある節点 r_i から節点 k を通過して，1-終端節点に到着するパスで表現される，アイテムベクトルと $\mathbf{q}$ との積のうち，$\eta(v_k)$ 以降の成分についての乗算の結果が格納されていると解釈することができる．

図 4 のアルゴリズムによる行列とベクトルの積の計算は，ZDD のノード数 $B(f_{\mathbf{X}})$ に比例する回数だけステップ 6 または 8 を実行することで実行することができる．疎行列とベクトルの積を求めるためには，一般には疎行列中の非零成分の個数分だけ演算を行う必要があるが，ZDD を用いることによって演算回数を ZDD のノードの個数分とすることができる．これにより ZDD による構造の圧縮が効果的に働く場合，計算回数を削減することができる．

図 3(b) の ZDD を用いて，$\mathbf{q} = \{1, 1, 1\}$ が与えられたときのアルゴリズムの挙動を示す．e_3, e_2, e_1 のそれぞれのラベルを持った節点について，value が $1, 1, 2$ と計算され，最終的に $\mathbf{s} = \{2, 1, 1\}$ が出力される．

Input: 実数ベクトル $\mathbf{q}$, 行列 $\mathbf{X}$ を表す ZDD $f_{\mathbf{X}}$.
Output: スコアベクトル $\mathbf{s}$

1: **for** $k \in \{1, 2\}$ **do**
2: value(k) $\leftarrow 0$ // 初期化
3: **end for**
4: **for** $k = 3$ to $B(f_{\mathbf{X}})$ **do**
5: **if** $v_k \in \{e_1, \ldots, e_M\}$ **then**
6: value(k) $\leftarrow$ value(l_k) + value(h_k) + $q_{\eta(v_k)}$
7: **else**
8: $s_{\eta(v_k)} \leftarrow$ value(h_k)
9: **end if**
10: **end for**
11: **return** $\mathbf{s}$

図 4: スコア計算手順

5. 行分割による節点数削減

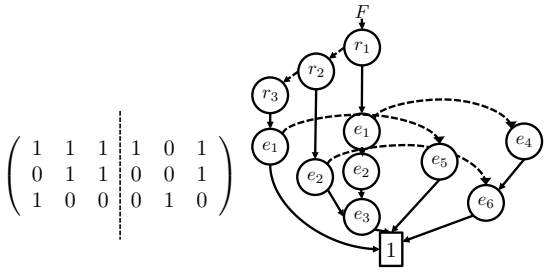
前節で導入した ZDD による 2 値行列の表現手法では，アイテムの特徴ベクトルを末尾から順に見ていってすべての要素が同一になる範囲でしか構造の共有は行われなかった．末尾から m 個の成分をとった m 次元ベクトルは 2^m 種類存在するため，m の値が大きくなるにつれ，末尾から M 個の要素がすべて等しい特徴ベクトルの個数は急激に減少し，節点の共有による圧縮が効かなくなる．そこで，構造を共有することによる圧縮をさらに活用するために，末尾以外が同一な場合でも構造の共有が行われるよう ZDD による表現を工夫する．具体的には，これまで 1 つの項によって表現していたアイテムを，アイテムを L 個の要素ごとに分割して作った項の集まりによって表現する．すなわち，(13) を，

$$Z_{\mathbf{X}} = \bigcup_{i=1}^{N} \bigcup_{j=1}^{\lceil M/L \rceil} \{r_i e_{a_{ij}(1)} \ldots e_{a_{ij}(b_{ij})}\} \qquad (14)$$

として表す．ここで $a_{ij}(k)$ は $\mathbf{x}_i$ の $(j-1)L+1$ から jL 番目の成分のなかで値が 1 となる k 番目の要素の添字を示す．b_{ij} は $\mathbf{x}_i$ の $(j-1)L+1$ から $\max(jL, M)$ 番目の要素のなかで 1 となる成分の個数を示す．図 5 は 3 つの 6 次元ベクトルからなる行列について，$L = 3$ として分割を行ったときに生成される ZDD を示している*1．分割を行わなかった場合にはラベル e_6 をもつ節点でしか共有は行われなかったが，分割することによって e_3 ラベルをもつ節点でも共有が起きていることがわかる．これは，分割によってあたかも 2 つの 3×3 の行列を ZDD で表現するような形になり，それぞれの行列について末尾の要素が等しいアイテム同士で共有が起きるためである．

一見，L を小さな値に設定するほど節点の共有による圧縮が起こるように見えるが，分割にはコストが伴う．図 5(b) の 1,2,3 列を 1 つの行列として考えると，1 行目と 2 行目では e_2 と e_3 が等しいため，共有が起こるべきである．しかし，4,5,6 列の行列へのリンクを 0-枝として保持しなければならないため，e_2 での共有が行われなくなっている．このように，0-枝が 0-終端節点の節点に繋がっている，ラベルが $e_1, \ldots, e_M$ であるような節点を以下では**先頭節点**とよぶ．先頭節点は 0-枝が終端節点以外の節点につながっているため共有による圧縮が起こりにくい．分割によって作られる各部分行列ごとに先頭節点が必要となるため，L が小さな値だと先頭節点が多く作ら

*1 図が複雑になるのを防ぐため，図 5(b) では 0 終端節点に達するリンクは省略している．



(a) 2値行列の例　　(b) 対応するZDD

図 5: 行分割の例

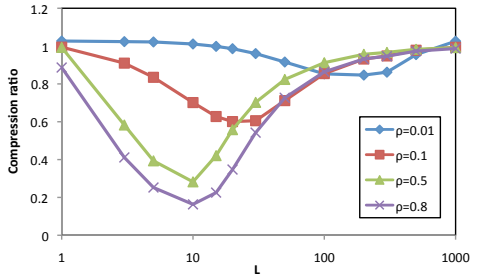


図 6: 分割数を変化させたときのランダム行列の圧縮率の変化

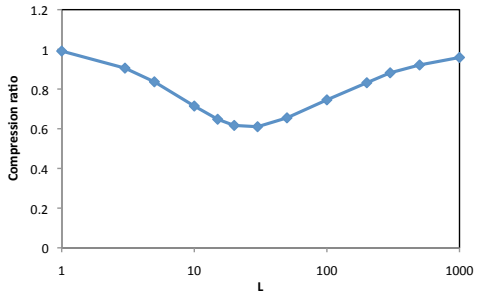


図 7: NIPS データセットの圧縮率の変化

れ，共有が起こりにくくなるという性質がある．そのため，L はデータに合わせた適切な値に設定する必要がある．

6. 検証

提案手法による演算回数の削減効果を 2 種類のデータを用いて検証する．まず人工的に大きさ 1000×1000 の 2 値行列を生成し，提案手法によってどの程度計算回数を削減できるかを調べる．ただし生成する行列は，各成分が Bernoulli 分布 $p(x_{ij}|\rho) = \rho^{x_{ij}}(1-\rho)^{1-x_{ij}}$ に従って生成されるものとした．次に [Ghahramani 06] の検証で用いられている NIPS データセットに適用し，計算量が削減される様子を調べた．NIPS データセットは国際会議 NIPS(Neural Information Processing Systems) の記事からなるデータセットであり，これに [Ghahramani 06] と同様の前処理を行い，2037 の著者と，その著者の記事に含まれる 13649 語からなる行列を作成した．ここでアイテムは各語に相当し，2037 次元のベクトルとして表現される．組み合わせ集合から対応する ZDD を構築する際に，湊らが開発した ZDD 処理系である VSOP[湊 05] を用いた．

まず，人工データについて，分割数を変化させたときの ZDD による圧縮率の変化を 6 に示す．図の横軸は L の大きさ，縦軸は圧縮率を表す．なお，圧縮率は，2 値行列中の値が 1 をとる成分の個数と，ZDD のノード数の比としている．これは，疎行列を表現する典型的なデータ構造では，疎行列中の 1 の個数に比例する記憶容量および計算を必要とするからである．図より，分割を行うことによって，最大で 20%から 80%程度の圧縮が可能なことが確認できる．特に ρ の値が大きな場合，すなわち行列中の 1 の出現率が高い場合に高い圧縮率を示すことが伺える．これは，ZDD 表現による圧縮の効率は，アイテム間で成分が 1 である共通な要素の個数に関係するため，ρ が大きいほどそのような共通な要素が増えるためである．また，圧縮率のピークは，ρ が大きいほど L が小さなときに出現する傾向があることが分かる．これは L 個の成分中の 1 の出現率に小さいと，その区間での先頭節点の割合が高まるため，圧縮が行われなくなるためと考える．

NIPS データセットに適用した結果を図 7 に示す．図より $L = 30$ のときに最大で約 60%の圧縮率を達成できることが分かる．なお，NIPS データの非零成分の出現率は約 0.04 であり，圧縮率が最大となる L は図 6 の $\rho = 0.1$ と $\rho = 0.01$ の圧縮率が最大となる L の間に位置する．この結果より，圧縮率を最大にする L を非零成分の出現率によって求められる可能性が示唆される．

7. おわりに

本稿では，集合拡張の代表的な手法である Bayesian Sets アルゴリズムについて，その計算量を削減する手法を示した．手法のポイントは，2 値行列とベクトルとの積算に必要な計算量を，2 値行列を ZDD を用いて圧縮した形式で表現することで削減したこと，および ZDD の構造を利用した演算によって積算を実現したことにある．また，節点数を削減することが可能な行列の ZDD による表現方法についても本稿で検討した．疎行列を単純に保持する手法に比べると，ZDD の構築には計算量が必要となるが，Bayesian Sets では一度生成した ZDD を繰り返し利用することができるため，構築にかかるコストを上回る利便性があると考える．今後は，行列の最適な分割を与える手法を検討する予定である．

参考文献

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

[Ghahramani 06] Ghahramani, Z. and Heller, K. A.: Bayesian Sets, in *In* Proc. of NIPS'06 (2006)

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

[Vickrey 10] Vickrey, D., Kipersztok, O., and Koller, D.: An Active Learning Approach to Finding Related Terms, in *In* Prox. of ACL'10, pp. 371–376 (2010)

[Wang 07] Wang, R. C. and Cohen, W. W.: Language-Independent Set Expansion of Named Entities using the Web, in *In* Proc. of ICDM'07, pp. 342–350 (2007)

[湊 05] 湊 真一：VSOP:ゼロサプレス型 BDD に基づく「重みつき積和集合」計算プログラム, 信学技報 COMP2005-10, Vol. 105, No. 72, pp. 31–38 (2005)