データベース解析のためのゼロサプレス型BDDの変数順序づけ方法とその評価

岩崎　玄弥†　　湊　　真一†　　ツォイクマン　トーマス†

† 北海道大学 大学院 情報科学研究科
E-mail: †{iwsk,minato,thomas}@ist.hokudai.ac.jp

あらまし　近年，ゼロサプレス型二分決定グラフ (ZBDD: Zero-suppressed Binary Decision Diagrams) を用いたデータベースの効果的な解析手法が提案されている．二分決定グラフ (BDD) は大規模論理関数データの表現方法として広く用いられている．今回はトランザクションデータベースに関して，大規模なアイテムの組合せ集合を処理するのに適した ZBDD を用いる．ZBDD のデータ構造は変数の順序に大きく影響を受ける．本稿では，大規模なトランザクションデータベースを ZBDD で表す際の新しい変数の順序づけ方法を提案し，その効果を評価した．
キーワード　データマイニング，　ZBDD，　頻出パタン

A Method of Variable Ordering for Zero-suppressed BDDs in Data Mining Applications

Haruya IWASAKI†, Shin-ichi MINATO†, and Thomas ZEUGMANN†

† Graduate School of Information Science and Technology, Hokkaido University
E-mail: †{iwsk,minato,thomas}@ist.hokudai.ac.jp

Abstract　Recently, an efficient method of database analysis using Zero-suppressed Binary Decision Diagrams (ZBDDs) has been proposed. BDDs are graph-based representation of Boolean functions, now widely used in system design and verification area. Here we focus on ZBDDs, a special type of BDDs, which are suitable for handling large-scale combinatorial itemsets in transaction database. The ZBDD size greatly depends on variable ordering. In this paper, we propose a new method of ZBDD variable ordering for the itemset mining of the large-scale transaction database, and show the experimental results.
Key words　database, ZBDD, frequent pattern

1. はじめに

近年，大規模記憶装置の発展などによって，大規模なデータベースの中から有用な規則を発見するデータマイニングの研究が盛んになっている．頻出パタンマイニング (Frequent Pattern Mining) は，最も基本的なデータマイニング問題であり，Agrawal 等 [1] による Apriori アルゴリズムの研究に始まり，現在までに様々なアルゴリズムが提案されている [5], [14]．

我々はこれまでに，VLSI CAD の分野で大規模論理関数データの表現法として広く用いられている二分決定グラフ (BDD: Binary Decision Diagrams) [2]，その中でも「ゼロサプレス型 BDD」(ZBDD: Zero-suppressed BDD) [9] と呼ばれるデータ構造を用いて，トランザクションデータベースにおける頻出パタンを効率よく生成する手法 [11], [12] に関する研究を進めている．ZBDD は大規模な組合せ集合データを非明示的に列挙し，頻出パタンの発見から解析に至る多様な演算を効率よく実行することができると期待されている．

ZBDD のグラフの大きさは，同じ例題に対しても変数の順序によって大きく影響を受けることが知られている．一般的な BDD の変数順序づけ方法については過去に多くの研究 [4], [6], [8] があるが，ZBDD を用いたデータベース解析処理におけるアイテム変数の順序づけに関しては，まだ良い方法は知られていない．本稿では，データベースの特徴に基づいて順序づけを行う新しい発見的手法を提案し，その効果を示す．

2. ZBDD によるデータベース表現

2.1 BDD

BDD [2] は，図 1 に示すような論理関数のグラフによる表現である．一般に，論理関数のそれぞれの変数に 0,1 の値を代入した結果を，二分岐の枝（0-枝/1-枝）で場合分けし得られる論

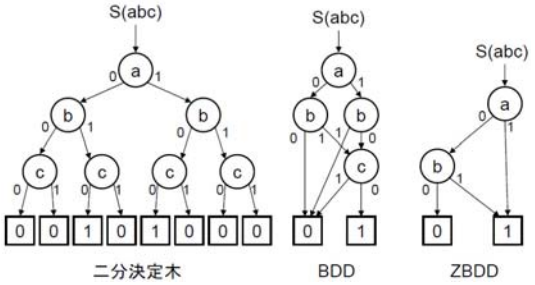


図 1　二分決定木と BDD,ZBDD

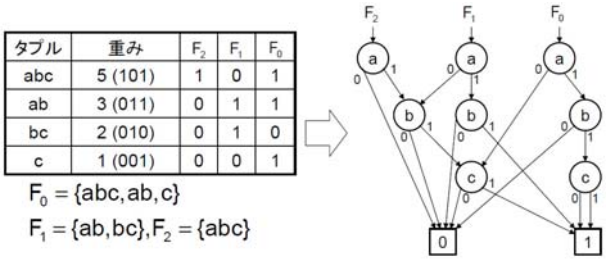


図 2　$(5abc + 3ab + 2bc + c)$ の 2 進ベクトルによる表現

理関数の値を，2 値の定数節点（0-終端節点/1-終端節点）で表現すると，図 1 のような二分木状のグラフになる．このとき，場合分けする変数の順序を固定し，冗長な節点の削除と等価な節点を共有するという 2 つの縮約規則を可能な限り適用することにより,「既約」な形が一意に得られることが知られている．複数の論理関数を表す BDD の間においても，変数順序を固定すればグラフを共有することができる．

BDD は，多くの実用的な論理関数を比較的少ない記憶量で一意に表現することができる．また，2 つの BDD を入力とし，それらの二項論理演算の結果を表す BDD を直接生成するアルゴリズムが考案されている．このアルゴリズムはハッシュテーブルを巧みに用いることで，データが計算機の主記憶の範囲に収まる限りは，その記憶量にほぼ比例する時間内で論理演算を効率よく実行できる．

2.2　ZBDD

BDD は元々は論理関数を表現するために考案されたものだが，これを用いて組合せ集合データを表現・操作することもできる．組合せ集合とは,「n 個のアイテムから任意個を選ぶ組合せ」を要素とする集合である．

これを BDD で表現するとき，類似する組合せが多ければ，部分的に共通する組合せがグラフ上で共有されて，記憶量や計算時間が大幅に削減される場合がある．さらに，組合せ集合に特化した ZBDD [9] を用いると，より簡潔な表現が得られ，一層効率よく扱うことができる．

ZBDD では，冗長な節点を削除する簡約化規則が通常の BDD とは異なる．ZBDD では 1-枝が 0-終端節点を直接指している節点を取り除く，という規則になっている．これにより，ZBDD では図 1 のように，組合せ集合に一度も選ばれることのないアイテムに関する節点が自動的に削除されることになり，BDD よりも効率よく組合せ集合を表現・操作することができる．

2.3　VSOP プログラム

VSOP(Valued-Sum-Of-Products calculator) [10] は，数式により組合せ集合の演算を記述し，これを ZBDD を用いて計算するプログラムである．VSOP は非常に多数のアイテム変数の組合せを要素とする積和集合を扱うことができ，また，単なる集合演算だけでなく，集合の各要素に整数値（係数あるいは重み）を定義し，加減乗除や大小比較などの算術演算を含む数式を処理できることを特徴とする．

ここで，ZBDD を用いて重み付き積和集合を非明示的に表現する方法について述べる．ZBDD は組合せ集合を表すことしかできないため，そのままでは重みを表現できない．これの解決法として，本稿では整数値を 2 進数に分解する方法を用いる．n 個の ZBDD$\{F_0, F_1, \ldots, F_{n-1}\}$ をベクトル状に並べると，最大 $(2^n - 1)$ までの重みを表現することができる．すなわち，整数値の重みを 2 進数で符号化し，最下位ビットが 1 になる組合せ集合を F_0，次のビットが 1 になるような組合せ集合を F_1，という形で F_{n-1} まで並べることにより，図 2 のように，各項の重みを非明示的に表すことができる．

2.4　ZBDD によるデータベース表現

VSOP を用いることで，トランザクションデータベースにおいて頻度表の生成を効率よく行うことができる．ここでは，重み付き積和集合表現を用いた「タプル頻度表」と「パタン頻度表」について述べる．

タプル頻度表とは，トランザクションデータ中に同じアイテムの組合せ（タプル）が何回出現したかを数えて表にしたものであり，パタン頻度表とは，各タプルの中に出現するアイテムの部分集合（パタン）の出現回数を数えて表にしたものである．通常の組合せ集合では，組合せパタンの有無のみに着目し重複は考慮しないが，実際のデータベースでは同じタプルやパタンが複数回出現することがあり，その出現回数（頻度）を数えることが必要である．このような場合に重み付き積和集合を用いると頻度表を効率的に生成することができる．また，一般に k 個のアイテムからなるタプルは 2^k 個のパタンを含んでいるので，パタン頻度表はタプル頻度表よりもはるかに大きなものになり，小規模な場合を除いて完全なパタン頻度表を生成することは困難である．しかし，ZBDD を用いる場合，多数の類似するパタンをグラフで共有してコンパクトに表現できるので，ある程度の規模までパタン頻度表を現実的に生成できる可能性がある．一旦パタン頻度表の生成に成功すれば，任意の頻度に対応する頻出パタン集合をすべて同時に保持しているのと同等であるため，非常に強力なデータ表現であり，その後は様々なデータベース解析処理を ZBDD 処理系の演算により効率よく解くことが可能となる．

2.5　ZBDD-growth 法による頻出パタン生成

パタン頻度表は生成できれば強力であるが，中規模以上のベンチマーク例題に対しては，ZBDD が巨大になり過ぎて，現実的な記憶量では表現することができない．しかし，ある一定の頻度以上の頻出パタン集合だけを抽出するだけであれば，最小頻度の閾値 α を大きくしていけばパタン数が単調減少するので，適度に大きい α を与えれば ZBDD を構築することが可

能となる．そこで我々の研究グループでは，ZBDD のベクトルで表現されたタプル頻度表から，パタン頻度表を経由せずに，直接，頻出パタン集合を表す ZBDD を生成するアルゴリズム「ZBDD-growth 法」を開発した [12]．これにより，中規模以上のベンチマーク例題でも頻出パタン集合を表す ZBDD を生成できるようになったが，それでもやはり ZBDD が小さければ計算時間が少なくてすむため，良い変数順序づけを求めることは依然として重要な研究課題である．

3. データベース表現における ZBDD の変数順序づけ

前述のように，ZBDD を用いることで頻度表や頻出パタン集合をコンパクトに表現できる可能性があるが，データベースが大規模になると，ZBDD を生成することが困難になる．これを改善する手法として，本稿では,「アイテム変数の順序づけ」を考える．

ZBDD の性質上，最悪な順序づけを用いても節点数はパタンを羅列したときの総文字数を超えることがない [11] ため，総文字数が少なければどのような順序づけでも，ZBDD のサイズが膨らむことはない．このことから，アイテム変数の順序づけにより指数関数的な効果が表れるためには，最悪な場合が指数関数的な節点数になる必要があるので，データ中にはパタンを羅列したときの総文字数が非常に多くなければならないということがいえる．一般的に，パタンはアイテム数の指数個あるので，アイテム変数の順序づけにより，大きな ZBDD の簡単化効果が得られると期待できる．

本稿で使用した VSOP プログラムでは，データベースに含まれるアイテムをアイテム変数として最初に宣言する．宣言されなかった変数は，その変数が算術式中で使用されたときに，その場で新たに宣言したものとして，最上位に追加される．(以下では，この変数順序を「後出し上位順」と呼ぶ.) 我々がこれまでに提案した ZBDD によるデータマイニングの実験は，ほとんどがこの順序づけを用いている．後出し上位順は，順序づけに要する計算コストが不要であり，経験的には比較的良い順序を得られることが多い．しかし，この順序づけでは，同じ内容のデータベースでも，処理するレコードの順番によって変数の順序が大きく変わるため，結果が不安定で，例題によっては非常に悪い順序づけとなる場合がある．そこで我々は，与えられたデータベース例題の特徴に基づいて，より良い変数順序を安定的に求めるための発見的手法を提案する．

3.1 論理回路における BDD の動的重みづけ法

通常の BDD の変数順序づけ方法については，これまでに VLSI 論理設計の分野で多くの研究がある．ここでは，論理回路における BDD の「動的重みづけ法」[8] を紹介する．この方法は，BDD を用いて論理回路の各信号線の論理関数を BDD で表す場合に，回路の結線情報から関数の性質を予想し，最適に近い順序を求めるものである．

一般の BDD においては，グラフの大きさに影響する要素として，次の 2 つの性質が知られている [4]．

(1) 局所計算性のある入力の組は，なるべく近い順序にする．これにより，多くのサブグラフが共有できるという傾向がある．例えば 2 進数の加算器の論理回路では，対応するビット同士の変数を並べて配置すると良いことが知られている．

(2) 出力を制御する力の強い入力は上位に配置する．例えば，データセレクタの論理回路では，データ入力を上位に並べた場合は制御入力を上位に並べた場合に比べ，BDD のサイズが指数的に増大してしまう．これは，制御入力の値を先に決定すると，多くのデータ入力を無視できるという性質があるためと考えられる．

これら 2 つの性質を満たすような順序づけを行えばよいのであるが，実際には，2 つの性質が同時に現れて，互いに相反する順序づけを要求する場合があり，これを両立させて最適な順序を求めるのは難しい．最適な順序を求める問題は NP 完全であることが知られている [13]．また，どんな順序でもそれほど変化の見られない回路もあり，その場合には，順序づけを工夫してもあまり効果が得られない．

以上の考察に基づいて，回路の結線情報から変数の順序づけを行う発見的手法がいくつか知られているが，その中の 1 つとして「動的重みづけ法」が提案されている [8]．

動的重みづけ法では，まず前述の (2) の性質を満たすため，制御性の高い入力を見つける．これは，以下のような傾向を持つ．

- ファンアウト数が多い（出力に至るパスが多い）.
- 出力までの段数が少ない.
- 出力へ至るパス上のゲートのファンイン数が少ない.

この傾向を評価するために，次に示すような簡単な重みづけ法を用いる．

(1) 出力線の重みを 1 とする．

(2) 出力から入力に向かって重みを伝達させる．2 入力以上のゲートでは，出力線の重みをファンイン数で均等に分けて入力線に伝える．

(3) ファンアウトがある信号線で，複数のゲートから重みが伝わるときにはそれらを加える．

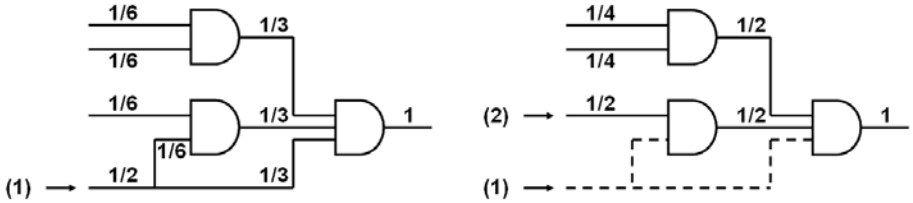
以上の手順で重みづけをした例を図 3(a) に示す．この重みが最大となった入力が最も出力を制御しやすいとみなし，その入力に最上位の変数を割り当てる．

次に，最上位の変数を決定した後，その下のサブグラフを小さくすることを考える．このとき，すでに決定された入力は 0 か 1 に固定されてしまっていると考えれば，その近くの信号線は，制御性が増大しているはずである．そこで，すでに決定された信号線を切断したと仮定して，改めて重みづけを行う．このように動的に重みづけを行うことにより，前節 (1) の局所計算性も反映することができる．その様子を図 3(b) に示す．以下，これを動的重みづけ法と呼ぶ.

動的重みづけ法では，順序づけに必要な時間は，入力数 n, 素子数を m として，O(n・m) 程度であるが，順序づけによって n の指数の違いが見られるので，それでも十分有効である．

3.2 トランザクションデータベースにおける ZBDD の順序づけへの応用

ここまで論理回路における BDD の順序づけ法に関して説明



(a) 順序づけ 1 回目　　(b) 順序づけ 2 回目

図 3　論理回路における動的重みづけ法

してきたが，本稿では，これをトランザクションデータベースにおける ZBDD の順序づけに応用する．

論理回路においては，出力から重みを伝えていきゲートの入力数に応じて重みを分配したが，本手法では，

(1) データベース全体の重みを 1 として，これをタプルの数で割った値をそれぞれのタプルの重みとして与える．

(2) 各々のタプルにおいてそのタプルに含まれないアイテムに関して，タプルに与えられた重みをその含まれないアイテムの数で割った値を重みとして伝える．

(3) 複数のタプルから重みが伝わるときにはそれらを加える．

の手順で重みづけを行う．

この (2) の手順において，タプルに含まれるアイテムではなく，タプルに含まれないアイテムに重みを伝えて行くところが，従来の論理回路での動的重みづけ法とは大きく異なる点である．タプルに含まれるパタンを抽出する場合に，タプルに含まれるアイテムはパタンを形成するアイテムの組合せに含まれるときと含まれないときがあるが，タプルに含まれないアイテムは当然そのタプルが含むパタンの要素とはなりえず，このことから，タプルに含まれないアイテムは含まれるアイテムより出力に与える影響が大きいと考えられるので，このように重みづけを行う．

このようにして重みづけをした例を図 4(a) に示す．この重みが最大となったアイテムを ZBDD のサイズに特に影響を与えるものとして，変数の最上位に割り当てる．与えられた重みが同じものが複数あった場合は，最も出現が早かったアイテムを選ぶ．

次に，最上位の変数を決定した後の処理について述べる．順序が決定したアイテムに関する線は BDD の場合と同様に取り除き，もしアイテムの順序づけが決まったことで，タプルからアイテムに伸びる線の全てが無くなってしまったタプルがあれば，そのタプルは以降の重みづけの際には存在しないものとして扱う．つまり，存在するタプルには，総タプル数から存在しなくなったタプルの数を引いた数で重み 1 を割った値を重みとして与えることになる．これを図 4(b) に示す．この順序が決定したアイテムに関する線を取り除くという処理によって，取り除かれたアイテムの重みがそのアイテムと関係の深いアイテムに分配されることになるので，局所計算性を反映することができる．

以上の操作を繰り返すことでアイテム変数の順序づけを行う．

4. 実験と考察

本実験において使用した PC は Pentium4，3.0GHz，SuSE Linux 9.3，主記憶 512Mbyte で ZBDD の最大節点数は 1,000 万個とした．はじめに，人工的な例題における ZBDD の順序づけ法の効果に関して行った実験の結果を示し，次に実際のベンチマーク例題における ZBDD の動的重みづけ法に関して実験を行った結果を示す．今回の実験で用いた順序づけは，動的重みづけ法，出現頻度の高いアイテムを上位に並べた「高頻度上位順」，逆に出現頻度の低いアイテムを上位に並べた「低頻度上位順」と，アイテムの最初の出現が遅いものから上位に並べた「後出し上位順」の 4 つである．

4.1 人工的な例題におけるアイテム変数の順序づけ法の評価

まず，VSOP のアイテム変数の順序づけ法に関して，人工的な例題における動的重みづけ法の効果を調べる．今回は以下に示すような人工的に生成したデータベースを用いて実験を行う．

$a2$	$a3$	$a4$	$\cdots$	an	$b2$	$b3$	$b4$	$\cdots$	bn
$a1$	$a3$	$a4$	$\cdots$	an	$b1$	$b3$	$b4$	$\cdots$	bn
$a1$	$a2$	$a4$	$\cdots$	an	$b1$	$b2$	$b4$	$\cdots$	bn
	$\vdots$				$\vdots$				$\vdots$
$a1$	$a2$	$a3$	$\cdots$	$a(n-1)$	$b1$	$b2$	$b3$	$\cdots$	$b(n-1)$

このデータは，$a1\ a2\ ...\ an\ b1\ b2\ ...\ bn$ という形から ak と $bk(k = 1, ..., n)$ を取り除いたものを各レコードとしている．また，このデータベースに関して動的重みづけ法を用いた場合のアイテム変数の順序づけと，低頻度上位順に並べた順序づけを以下に示す．

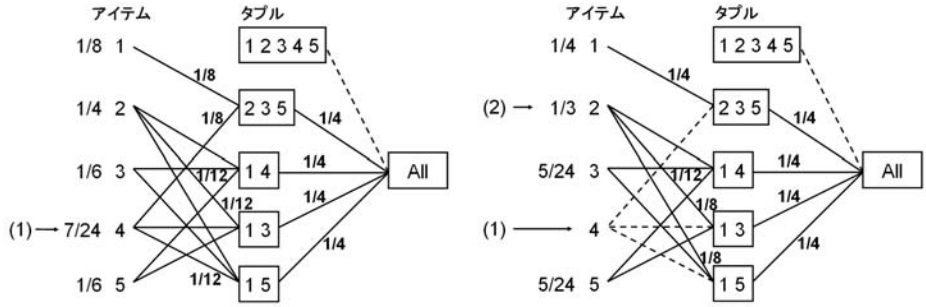
動的重みづけ法: $a2\ b2\ a3\ b3\ a4\ b4\ \cdots\ an\ bn\ a1\ b1$

低頻度上位順: $a2\ a3\ a4\ \cdots\ an\ b2\ b3\ b4\ \cdots\ bn\ a1\ b1$

今回は低頻度上位順を用いたが，高頻度上位順や後出し上位順でも低頻度上位順とほぼ同じ順序づけとなった．

以上の場合において，タプル頻度表から頻出パタンを抽出する実験を行った結果を表 1 に示す．

この結果を見ると，アイテム変数の順序づけが ZBDD のサイズに大きな影響を及ぼすことがわかる．動的重みづけ法の場合はアイテム変数の数に比例して ZBDD のサイズが増加しているのに対して，低頻度上位順では指数的にサイズが増加している．これは，今回の人工的なデータベースでは全てのアイテムの出現回数が同じであるので，出現回数で順序づけをしよう



(a) 順序づけ 1 回目　　(b) 順序づけ 2 回目

図 4　トランザクションデータベースにおける動的重みづけ法

表 1　人工的な例題における VSOP の処理時間

	動的重みづけ法		低頻度上位順	
n	size	time(s)	size	time(s)
2	5	<0.1	5	<0.1
3	10	<0.1	12	<0.1
4	15	<0.1	25	<0.1
5	20	<0.1	50	<0.1
6	25	<0.1	99	<0.1
7	30	<0.1	196	<0.1
8	35	<0.1	389	<0.1
9	40	<0.1	774	<0.1
10	45	<0.1	1,543	<0.1
12	55	<0.1	6,153	<0.1
14	65	<0.1	24,587	<0.1
16	75	<0.1	98,317	<0.1
18	85	<0.1	393,231	0.456
20	95	<0.1	1,572,881	1.954
22	105	<0.1	6,291,475	8.482
24	115	<0.1	Memory overflow	

としても結局アイテムの出現順に並んでしまうため，低頻度上位順では効果的な順序づけができなかったためである．これに対して動的重みづけ法では，1 つのアイテムの順序が決まったときに，その決まったアイテムの重みが関連の深いアイテムに分配されることになるので，ZBDD のサイズに関して非常に縮約効果のある順序づけができたのだと考えられる．このことから，この人工的な例題では，局所計算性が大きく影響したと考えられる．

このように，アイテムの出現頻度だけでは効果的な順序づけができない場合においても，動的重みづけ法では効果的な順序づけができることが示された．

4.2　動的重みづけ法の効果

次に，実際のベンチマーク例題における ZBDD の動的重みづけ法による順序づけの効果に関して行った実験の結果を示す．実験は，動的重みづけ法を用いてタプル頻度表を構築する

表 3　動的重みづけ法の順序づけの計算時間

chess	2.8sec
mushroom	17.9sec
connect	219.6sec
BMS-WebView-1	1,255.0sec

VSOP スクリプトを生成し，そこから ZBDD-growth 法を用いて頻出パタンを抽出するという形で，生成される ZBDD の節点数を調べた．その結果を表 2 に示す．データベース名の隣の () 内の数字は，抽出するパタンの最低頻度を表したものである．つまり，この数字より頻度が高いパタンを抽出するということになる．

この実験結果より，今回の動的重みづけ法はアイテム変数を低頻度上位順に並べたものと非常に近い効果を得られることがわかった．BMS-WebView-1 に関しては今回の手法の方がよい結果を得られている．また，高頻度上位順と比べると大きな縮約効果が得られていることがわかる．

また，動的重みづけ法にかかる処理時間は表 3 のようになる．

4.3　考　　察

今回の実験で，ZBDD における動的重みづけ法の順序づけによる ZBDD の縮約効果が確認できた．出現頻度だけでは効果的な順序づけができない場合にも，動的重みづけ法は効果的な順序づけを行えることがわかった．しかし，実際のベンチマーク例題において実験を行った場合では，高頻度上位順と後出し上位順よりは効果的な順序づけができていることが確認できたが，低頻度上位順とは大きな差は見られなかった．これは，人工的な例題では局所計算性が非常に大きく影響したが，実際のベンチマーク例題ではそれほど大きく影響しなかったためと思われる．また，低頻度上位順では出現頻度の低いアイテムから上位に並べられるが，これは動的重みづけ法で出現頻度の低いアイテムにより大きな重みが伝えられる傾向があるということと類似している．このことから，実際のベンチマーク例題においては動的重みづけ法と低頻度上位順では似たような順序づけになったと考えられる．

このように，局所計算性よりもアイテムの出現頻度の方が ZBDD のサイズに大きな影響を与える場合においては，順序づけにかかる計算時間が短い出現頻度に関する順序づけ法の方

表 2　トランザクションデータベースにおける ZBDD の動的重みづけ法の効果

	動的重みづけ法		高頻度上位順		低頻度上位順		後出し上位順	
	size	time(s)	size	time(s)	size	time(s)	size	time(s)
chess (2,000)	1,422	5.8s	3,856	2,036.6s	1,415	5.8s	2,778	64.8s
mushroom (1)	16,403	1.0s	448,734	1.9s	15,131	1.0s	40,557	1.1s
connect (60,000)	348	27.5s	1,659	5,402.3s	348	27.5s	374	62.8s
BMS-WebView-1 (30)	106,920	98.8s	389,181	778.2s	109,989	103.1s	152,431	153.4s

が効果的なこともある．例えば，動的重みづけ法の順序づけに BMS-WebView-1 では 1,255.0 秒かかっているが，単純に頻度を計算するだけなら 0.8 秒ですむ．このことから，プログラミングにはまだ改善の余地はあるが，低頻度上位順で十分な縮約効果が得られるならば，低頻度上位順で順序づけを行ってもよいといったことがいえるのかもしれない．これは議論すべき課題の一つであるといえる．

また，実験では ZBDD-growth 法を用いてタプル頻度表から頻出パタンを抽出したが，その際にデータベースの種類やアイテム変数の順序づけによって，ZBDD の生成にかかる時間が著しく変化することがわかった．例えば，chess では動的重みづけ法では 5.8 秒ですんだが，高頻度上位順では 2,036.6 秒もかかった．しかし，mushroom においてはどの順序づけでも大きな処理時間の差はなかった．

5. おわりに

本稿では，データベース解析において生成される ZBDD の簡単化に関して種々の実験を行った．その結果，トランザクションデータベースにおける ZBDD の動的重みづけ法による ZBDD の縮約効果が確認できた．人工的な例題においては局所計算性を反映することで，効果的な順序づけを行うことができた．また，実際のデータベースにおいても効果的な順序づけを行えることが確認できたが，出現頻度をもとに行った順序づけと大きな差が生まれない場合もあるということがわかった．今後は動的重みづけ法の高速化を行い，様々なデータベースに関して実験を行うとともに，データベースの特徴が順序づけの効果にどのように影響するのかも調べたい．

謝　　辞

本研究の一部は日本学術振興会科研費補助金 基盤 (B) 「二分決定グラフに基づく大規模データベースの効率的解析処理アルゴリズムの研究」(課題番号 17300041, 研究代表者 湊 真一) による．

文　　献

[1] R. Agrawal, H. Mannila, R. Strikant, H. Toivonen and A. I.. Verkamo, Fast Discovery of Association Rules, In Advances in Knowledge Discovery and Data Mining, MIT Press, 307-328, 1996.

[2] Bryant, R. E., Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput., C-35, 8 (1986), 677-691.

[3] M. Fujita, H. Fujisawa and N. Kawato. Evaluation and improvement of Boolean comparison method based on binary decision diagrams. In Proc. of IEEE/ACM International Conference on Computer-Aided Design (ICCAD-88), pp. 2-5, Novenber 1988.

[4] Fujita, M., Fujisawa, H., and Kawato, N., Evaluation and Implementation of Boolean comparison method based on binary decision diagrams, In Proc. ACM/IEEE International Conf. on Computer-Aided Design(ICCAD-88), (1988), 2-5.

[5] J. Han, j. Pei, Y. Yin, R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowkedge Discovery, 8(1), 53-87, 2004.

[6] Malik, S., Wang, A. R., Brayton, R. K., and Vincentelli, A. S., Logic verification using binary decision diagrams in a logic synthesis environment, In Proc. ACM/IEEE Internationl Conf. on Computer-Aided Design(ICCAD-88), (1988), 6-9.

[7] S. Minato, N. Ishiura and S. Yajima. Shared binary decision diagram with attributed edges for efficient Boolean function manipulation. In Proc. of 27th ACM/IEEE Design Automation Conference (DAC'90), pp. 52-57, June 1990.

[8] 湊 真一, 石浦 菜岐佐, 矢島 脩三, 論理関数の共有二分決定グラフによる表現とその効率的処理手法, 情報処理学会論文誌, Vol.32, No.1, pp.77-85, January 1991.

[9] Minato, S., Zero-suppressed BDDs for set manipulation in combinatorial problems, In Proc. 30th ACM/IEEE Design Automation Conf. (DAC-93), (1993), 272-277.

[10] 湊真一, VSOP: ゼロサプレス型 BDD に基づく「重み付き積和集合」計算プログラム, IEICE Technical Report COMP2005-13(2005-5)

[11] 湊真一, 有村博紀, ゼロサプレス型二分決定グラフを用いたトランザクションデータベースの効率的解析手法, 電子情報通信学会論文誌, Vol.J89-D, No2, pp. 172-182, 2006.

[12] S. Minato, H. Arimura. Frequent Pattern Mining and Knowledge Indexing Based on Zero-suppressed BDDs. In Proc. The 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID-2006), pp. 83-94, Sep. 2006.

[13] Tani, S., Hamaguchi, K., and Yajima, S., The complexity of the optimal variable ordering problems of shared binary decision diagrams, In 4th International Symposium on Algorithms and Computation, LNCS-762, Springer(1993), 389-398.

[14] M. J. Zaki, Scalable Algorithms for Association Mining, IEEE Trans. Knowl. Data Eng. 12(2), 372-390, 2000.