F-012

頻出パタン集合を表現する二分決定グラフの変数順序改善法に関する実験と考察
Experiments and Considerations on Variable Ordering of BDDs for Representing Frequent Patterns

礒松 紘平† 湊 真一† ツォイクマン トーマス†
Kouhei Isomatsu Shin-ichi Minato Thomas Zeugmann

1. はじめに

近年，大容量記憶装置の急速な発展等によって，大規模なデータ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．その中でも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[6] 等による 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．

本稿では，VLSI CAD の分野で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二分化決定グラフ（BDD：Binary Decision Diagrams）[1]，その中でも「ゼロサプレス型 BDD（ZBDD：Zero-suppressed BDD）[7], [8] と呼ばれるデータ構造を用いて、頻出パタン集合を表現する場合の ZBDD の変数順序改善法に関する実験と考察を行った．

これまで、ZBDD におけるアイテムの順序付けが，生成される ZBDD の大きさに多大な影響を与えることがわかっており、すでに岩崎らによるヒューリスティックな ZBDD の変数順序付けプログラム [5] が考案されているが，それがどれ位最適解に近いのかはっきりとは判っていない．そこで本実験では，生成された ZBDD のアイテム同士を交換する等の手法により、どの程度まで ZBDD の大きさを改善できるか調べた．

2. ZBDD によるデータベース表現

2.1 BDD

BDD 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ついて，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の共有という 2 つの縮約規則を可能な限り適用することにより，「規約」な形が得られ，論理関数をコンパクト且つ一意に表せ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可能である．

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 [1] 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．

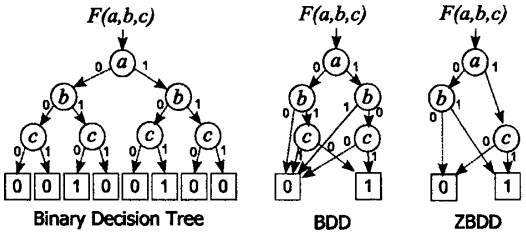

図 1: 二分決定木と BDD，ZBDD

ID	タプル		
1	abc		
2	ab	→	最小頻度 $\alpha = 10$:$\{b\}$
3	abc		
4	bc	→	最小頻度 $\alpha = 8$:$\{a, b, c\}$
5	ab		
6	abc	→	最小頻度 $\alpha = 7$:$\{ab, bc, ac, a, b, c\}$
7	c		
8	abc	→	最小頻度 $\alpha = 5$:{abc,ab,bc,ac,a,b,c}
9	abc		
10	ab	→	最小頻度 $\alpha = 1$:{abc,ab,bc,ac,a,b,c}
11	bc		

(データベース例題)

図 2: データベースの頻出パタン集合

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，これを用いて組み合わせ集合データを表現・操作することも出来る．組合せ集合とは，「n 個のアイテムから任意個を選ぶ組み合わせ」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「ゼロサプレス型 BDD（ZBDD）」[7] 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出来る．

ZBDD では，冗長な節点を削除する簡約化規則が通常の BDD と異なり、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・操作することが出来る．

†北海道大学大学院情報科学研究科

2.3 頻出パタン集合の ZBDD 表現

頻出パタン集合の ZBDD 表現とは，最小頻度 α を与えたときに，データベース中に α 回以上出現するアイテムの部分集合（パタン）を列挙し，ZBDD を使って主記憶上でコンパクトにそれを表現し，高速に計算を行うという手法である．ZBDD は組合せ集合をコンパクトに圧縮して表現するだけでなく，様々な集合演算を効率よく実行できるという特徴を持ち，主記憶上で大規模な組合せ集合データを扱うことが出来るため，最近注目されている．ここで図 2 に示すように，あるデータベース例題に対して，最小頻度 α 回以上出現するアイテムを考えると，例えば $\alpha = 7$ の時，頻出パタンは {ab,bc,ac,a,b,c} となる．本稿では，このアイテムの全ての頻出パタン集合を表す ZBDD を直接生成する，湊らによる「ZBDD-growth 法」[10] というアルゴリズムを使用した．

3. 頻出パタン集合を表す ZBDD の変数順序改善

データベースが大規模になると、ZBDD を用いても頻出パタン集合を生成することが困難になる．ZBDD の変数順序を改善するに当たって，最初に変数の順序を決定することが必要であり，動的重み付け法と呼ばれるヒューリスティックな方法を用いた岩崎らの順序付け方法 [5] が既に考案されているが，この方法で得られた結果が最適解にどの程度近い物であるかはっきりとは判断できない．

そこで本稿では，ZBDD 作成後の変数交換という手法を用いて ZBDD のノード数の改善を試みた．それが「変数順序改善」である．この方法に類するものとして，過去に隣接変数の交換に基づく逐次改善手法 [2] やアニーリング法 [4] 等の手法が提案されている．具体的な方法として，ある変数の順序で ZBDD を作成し，そこから変数の入れ替えを行い，ZBDD のノード数を減らす，ということを行う．本稿では，変数の交換を行ったときに，どの程度まで改善できるのか，また，演算処理にかかる時間がどう変化するのかについて調べた．

3.1 変数の交換アルゴリズム

本実験ではまず，隣り合う 2 変数の交換について調べた．具体的には，n 個の変数について，i 番目の変数と，$i+1$ 番目の変数を交換する．交換前より ZBDD のサイズが小さくなればその交換を採用する．その交換を $i = 1 \sim n-1$ までの変数について繰り返し行い，1 回のループの中でサイズが小さくなる交換が見つからなくなれば終了するという手法である．この手法では元のデータより悪くなることはないが，局所的な最適解に捕まりやすいと思われる．そこで，隣同士の変数を入れ替えただけでは一定の局所最適解に陥りやすくなると考え，2 つ以上離れた変数同士の入れ替えに関しても調べた．具体的には i 番目の変数から距離 d だけ離れた変数同士について，$i = 1 \sim n-d$ までの交換を同様に繰り返し行い，その結果について，隣同士の交換とどのように異なるのか実験した．但し，今回の実験で距離 d は，例えば $d = 4$ とした時は，距離 4 だけ離れた変数の交換のみ行い，4 未満の距離の変数交換は行っていない．

4. 実験結果と考察

本実験において使用した PC は Celeron(R)，2.8GHz，SuSE Linux9.3，主記憶 512Mbyte で ZBDD の最大ノード数は 1000 万個とした．

4.1 ZBDD の変数順序改善

本手法を用いた ZBDD の変数順序改善のための実験として，代表的なトランザクションデータベース [3] について，まず初期順序をデータベース出現順とした ZBDD 上で隣り合う変数同士の交換を行い，ZBDD のノード数がどれだけ変化するのか，またその結果が岩崎らの変数順序付けプログラムによる順序から始めた場合ではどれ程の違いが生じるかを調べた．その結果を表 1 に示す．

ここで岩崎らによる変数の順序付けプログラムと照らし合わせるため，ZBDD-growth の最小頻度のしきい値 α の値は，岩崎らが行った研究時に使用した値と同じものになっている．ここで初期ノード数は交換を行う前のノード数，終了ノード数は交換により変化したノード数，d は交換する変数同士の距離，times(s) はプログラムの処理時間（秒）である．

この結果を見ると，データベースの出現順に ZBDD を作成した場合，改善後の ZBDD のノード数が初期ノード数から大きく改善されているが，岩崎らの順序付け法を用いて作成した ZBDD の場合は，初期ノード数自体が小さく，改善の度合いもデータベースの出現順の場合に比べて小さくなっているなっているものが多い．この事から岩崎らの変数順序付けプログラムがある程度最適解に近い順序を求められていることが判った．

また，処理時間に関してデータベースの出現順に比べて岩崎らの変数順序付けでは大きく減少しているが，これは ZBDD の演算がノード数にほぼ比例するため，初期ノード数の小さい岩崎らの変数順序付けの方が処理時間が大幅に短くなったものと思われる．

4.2 変数間の距離 d を変化させた場合の交換

次に補足的な実験として mushroom と chess ついて，距離 d を 1 から 10 まで変化させて同様の実験を行い，局所最適解がどの様に変化するかを調べた．その結果を表 2 に示す．

この結果を見ると，局所最適解に落ち込んでいるのはどの場合も同じでも，交換するアイテム間の距離によって，改善後の ZBDD のノード数の差が顕著に現れている場合があるが，どの d がよいというのは一概には言えず，扱うデータによって異なると思われる．今後も研究を続ける必要がある．

5. おわりに

本稿では，データベースの頻出パタン集合を表現する ZBDD の変数順序改善を行う方法について述べた．実験では，生成された ZBDD に関して，隣接した変数同士の入れ換えを行い，初期順序を順序付けされた ZBDD がどの程度最適解に近づいているかを調べた．その結果，変数の交換によって，生成された ZBDD がどのくらい改善の余地があるか，という目安を与えることが出来た．

また，岩崎らによる変数順序付けプログラムが，ある程度最適解に近い解を求められているという感触を得た．

表 1: 変数順序改善の効果

	初期順序:データベースの出現順			初期順序:岩崎らによる変数順序付け		
SampleData(α)	終了ノード数	初期ノード数	time(s)	終了ノード数	初期ノード数	time(s)
mushroom(1)	17,812	40,557	77.2	11,591	15,131	20.7
chess(2,000)	2,595	2,778	67.6	1,292	1,422	6.3
connect(15,000)	22,027	30,898	7,938.3	22,104	25,327	95.3
BMS-WebView1(30)	150,464	152,431	586.2	105,302	106,920	230.9

表 2: 距離 d を変化させた時の改善後のノード数の変化

mushroom ($\alpha = 1$)　初期ノード数:**40557**

d	ノード数	time(s)	d	ノード数	time(s)
1	17,812	77.2	6	11,661	20.9
2	12,567	42.3	7	13,324	20.0
3	11,840	30.6	8	11,818	18.0
4	11,496	29.5	9	13,430	11.7
5	11,645	23.3	10	12,867	12.5

chess ($\alpha = 2,000$)　初期ノード数:**2778**

d	ノード数	time(s)	d	ノード数	time(s)
1	2,595	67.6	6	1,687	68.0
2	2,621	67.9	7	1,802	68.1
3	2,495	68.1	8	1,836	68.0
4	2,133	67.8	9	1,865	69.0
5	1,833	68.1	10	1,877	68.2

アイテム間の距離 d を変化させた交換では，局所最適解に陥るのは同じでも得られる縮約効果についてはある程度の違いがあることが確認出来た．但し今回の方法では単純な交換規則でしか実験を行わなかったので，今後は局所最適解から抜け出す方法も含めて，考察を行っていく予定である．

謝辞

本研究の一部は，日本学術振興会科研費 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」（代表：湊真一，課題番号 17300041）による．

参考文献

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

[2] Fujita, M., Matsunaga, Y., and Kakuda, T., On variable ordering of binary descision diagrams for the application of multi-level logic synthesis, In Proc. IEEE European Design Automation Conf. (EDAC-91), (1991), 50-54.

[3] Goethals, B. and Zaki, M. J.: Frequent itemset mining dataset repository (2003), Frequent Itemset Mining Implementations (FIMI'03), http://fimi.cs.helsinki.fi/data/

[4] Ishiura, N., Sawada, H., and Yajima, S., Minimization of binary descision deagrams based on exchange of variables, In Proc. ACM/IEEE International Conf. on Computer-Aided Design(ICCAD-91), (1991), 472-475.

[5] 岩崎 玄弥, 湊 真一, ツォイクマン トーマス: "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," 電子情報通信学会 データ工学研究会 (DEWS 2007), 信学技報, DEWS2007 M4-6, Apr. 2007.

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

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

[8] Minato, S., Zero-suppressed BDDs and Their Applications, International Journal on Software Tools for Technology Transfar(STTT), Springer, Vol.3, No.2, pp. 156-170, May 2001.

[9] 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).

[10] 湊 真一: "データベースの頻出アイテム集合を表すゼロサプレス型 BDD の変数順序付けの理論的考察", 電子情報通信学会コンピュテーション研究会, 信学技報 Vol. 106, No. 566, COMP2006-55 pp. 37-42, Mar. 2007.