F-011

頻出パタンマイニングのためのゼロサプレス型 BDD の変数順序付け方法の高速化の検討

Consideration of Accelerating ZBDD Variable Ordering for Frequent Pattern Mining

岩崎 玄弥† 湊 真一† ツォイクマン トーマス†
Haruya Iwasaki Shin-ichi Minato Thomas Zeugmann

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 [9, 3]．

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)[5] 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 [6, 7] 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．我々は以前に，一般的な BDD の変数順序付け方法として湊等によって提案された「動的重み付け法」を，ZBDD の変数順序付け方法に応用する手法を提案した．この手法を用いることにより，比較的よい順序付けを行えることが確認できた．しかし，この動的重み付け法は，アイテム数を n，データベースのサイズを m とすると，$O(nm)$ の計算時間を必要とし，データベースが大規模になると現実的な時間内に変数の順序付けを行うことが困難になってしまう．本稿では，動的重み付け法を用いた変数順序付け法にかかる計算時間を短縮するために，データベースの一部をサンプリングして順序付けを行い，そのときの処理時間と順序付けの品質について評価した結果を示す．

2. ZBDD による頻出パタン集合表現

本稿では，以下に示すようなデータベースを考える．まず，M を空でない集合とする．M の要素をアイテムと呼ぶ．この集合から得られる全てのアイテムの組合せ集合は M のべき集合 $\wp(M)$ である．部分集合 $\mathcal{C} \subseteq \wp(M)$ は組合せ集合と呼ばれる．この組合せ集合の要素をタプルと呼ぶ．トランザクションデータベースはタプルのリストである．

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 [2] 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．

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)」[5] 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．

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

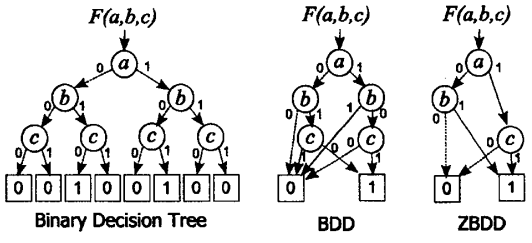

図 1: 二分決定木と BDD,ZBDD

†北海道大学大学院情報科学研究科

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: データベース例題とその頻出パタン集合

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の手法を用いることにした．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，湊等による「ZBDD-growth 法」[7] というアルゴリズムを使用した．

3. データベース表現における ZBDD の変数順序付け

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．これを改善するために，我々は「アイテム変数の順序付け」を行い ZBDD のサイズを小さくすることを考える．しかし，ZBDD の節点数を最小にする順序付けを求めることは NP 完全であることが示されている [8]．よって，我々の目的は ZBDD のサイズができるだけ最小に近くなる順序付けを行うことができる手法を開発することである．

我々は以前に，ZBDD の変数順序付け法として，トランザクションデータベースに対する動的重み付け法 [4] を提案した．この手法は，論理回路の分野で提案された通常の BDD の変数順序付け法である動的重み付け法をトランザクションデータベースにおける ZBDD の変数順序付けに応用したものである．

3.1 トランザクションデータベースにおける ZBDD の順序付けへの応用

まず初めにデータベースの構造を表すグラフを作る．各々のアイテムを表すノードを生成し，次に各々のタプルを表すノードを生成する．最後にデータベース全体を表すノード *All* を作る．さらに，ノード *All* から全てのタプルに対して枝を引き，各々のタプルから，そのタプルに含まれないアイテムに対して枝を引く．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．また，全てのアイテムに *All* に与える重みと同じ 1 を与える．

(2) 各々のタプルに含まれるアイテムに関して，タプルに与えられた重みをタプルに含まれないアイテムの数で割った重みをそれぞれのアイテムから引く．

(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．

の手順で重み付けを行う．

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の重みが相対的に増加するように重みを伝えてい行くところが，従来の論理回路での動的重み付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，このように重み付けを行う．

この重み付けの結果，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付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付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付けを行う．計算時間は，アイテム数を n，データベースのサイズを m とすると，$O(nm)$ となる．

4. サンプリングを用いた変数順序付け法の高速化

前述のようなトランザクションデータベースに対する動的重み付け法を用いることによって，比較的よい ZBDD の変数順序を求めることができることが示されている．しかし，$O(nm)$ の計算時間でも，データベースの規模が大きくなると現実的な時間内で順序を求めることは難しくなる．そこで本稿では，トランザクションデータベースをサンプリングし，そのサンプリングされたタプルのリストを用いて変数の順序付けを行うということを考える．この手法の目的は，データベース全体を用いた場合よりも変数順序の質は多少劣るかもしれないが，現実的な計算時間で変数の順序付けを行えるようにしようということである．この手法の有効性について行った実験の結果は後述する．

サンプリングにより得られるデータには，元のデータベースに現れる全てのアイテムが含まれていない場合がある．このため，一部のアイテムの変数順序は未決定となることがある．順序付けが未決定のアイテムが ZBDD の生成中に参照された場合には，そのアイテムは順序付けの最上位に配置される．サンプリングレートを 0%に設

表 1: トランザクションデータベースにおける ZBDD の動的重み付け法の効果

	100%		10%		5%		0%	
Database (min. support)	size	time(s)	size	time(s)	size	time(s)	size	time(s)
chess (1)	456,536	0.48s	456,801	0.08s	475,656	0.06s	1,029,615	0.0s
mushroom (1)	16,403	1.38s	17,092	0.15s	17,296	0.09s	40,557	0.0s
connect (15,000)	25,327	18.91s	25,296	1.96s	25,267	1.01s	(*)	0.0s
BMS-WebView-1 (30)	106,920	12.71s	108,189	0.99s	108,444	0.30s	152,431	0.0s

*: memory overflow

定した場合，全てのアイテムは元のデータベース中で最初に出現する位置が遅いものほど順序付けの上位に配置されることになる．この順序付けを我々は「後出し上位順」と呼び，以前にこの順序付けに関しても実験を行った．その結果，動的重み付け順ほどよくはないが，ある程度よい順序付けを行えることが確認された．今回の実験は，この後出し上位順と動的重み付け法を組み合わせた手法を提案しているとも言える．

5. 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 1GByte で最大節点数は 2,000 万個とした．今回の実験では，トランザクションデータベース全体を用いて重み付けを行った場合と，データベースからランダムに総タプル数の 0%，5%または 10%のタプルを抽出し，それを用いて重み付けを行った場合とを比較した結果を示す．0%というのは，重み付け法を用いた変数順序付けを行わないということと同義である．実験では，データベースに対してある割合でサンプリングを行い，変数の順序付けをし，ZBDD を生成するということを 20 回行い，その平均節点数と平均計算時間を計算し比較した．また，変数の順序付け法として動的重み付け法を用い，順序付けされていないアイテムが ZBDD の生成中に参照された場合には，そのアイテムは順序付けの最上位に配置される．

表 1 に実験結果を示す．size は ZBDD の平均節点数，time は動的重み付け法の平均計算時間である．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．この表を見ると，サンプリングレートが 5%や 10%とかなり小さい場合でも，データベース全体を用いた場合と比べて，ZBDD のサイズにあまり差がないことがわかる．一方で計算時間は，サンプリングレートに比例して減少していることがわかる．これにより，データベースの一部を用いるだけでも，比較的よい順序付けをデータベース全体を用いる場合と比べて高速に行えることがわかった．また，ZBDD の生成にかかる時間はサンプリングレートを変えてもほとんど変化しなかった．

6. 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 の変数順序付けの高速化に関して種々の実験を行った．その結果，サンプリングレートがかなり低い場合においても，データベース全体を用いた場合の変数順序付けの品質と比較して，あまり差が生まれないことがわかった．その一方で変数順序付けに要する計算時間はサンプリングレートに比例して減少することがわかった．これらのことから，実際にデータベース解析を行う場合には，その状況に応じてサンプリングレートを調節し，ZBDD を生成することが有効であると考えられる．今後は様々なデータベースに関して実験を行い，データベースの特徴が実験結果にどのような影響を与えるのかを調べたい．

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．

参考文献

[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] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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