D-031

頻出パタン抽出アルゴリズム「LCM over ZDDs」の変数順序付けの影響に関する考察

Considerations on Variable Ordering Effects for the Itemset Mining Algorithm LCM over ZDDs

岡崎 佑太 † 湊 真一 †
Yuta Okazaki Shin-ichi Minato

1. はじめに

近年，データから新たな知識を発見するデータマイニング技術の重要性が高まっている．その中でも頻出パタン抽出問題は最も基本的な問題として知られている．

本研究では，頻出パタン集合の表現に適した二分決定グラフ (BDD: *Binary Decision Diagrams*)，その中でもゼロサプレス型 BDD (ZDD: *Zero-suppressed BDD*) と呼ばれるデータ構造を用いる．頻出パタン列挙には LCM(*Linear time Closed item set Miner*) という高速なアルゴリズムがあり，これを利用し ZDD で解集合をコンパクトに出力する LCM over ZDDs が最近提案された．LCM では変数を展開する順序で効率が大きく変化する．元の LCM では低頻度上位順が良いとされているが，LCM over ZDDs でも同様なのか，実験を行いその傾向を調べた．

2. 頻出パタン抽出問題

本研究で扱うデータベースは，「タプル」と呼ばれるアイテムの組合せをリスト化したものである．また，このタプルに含まれるアイテムの部分集合を「パタン」と呼ぶ．頻出パタン抽出問題とは，データベース中に最小頻度 α 回以上出現するパタンを抽出・列挙する問題である．図 1 のようにパタンはタプルから導出することが出来るが，一般にアイテムが k 個あるとするとパタンは 2^k 通りに及ぶので，大規模なデータベースから頻出パタンを抽出することは容易ではない．

ID	レコード
1	a b
2	a
3	a c
4	b
5	a b c
6	a b
7	b
8	b c

頻出パタン集合

$\alpha=6 \longrightarrow \{b\}$
$\alpha=5 \longrightarrow \{a, b\}$
$\alpha=3 \longrightarrow \{a, b, c, ab\}$

図 1 データベースから頻出パタンを抽出

† 北海道大学大学院情報科学研究科

3. BDD/ZDD を用いたデータベース表現

BDD[1] とは論理関数をグラフによって表現したものである．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，二分岐の枝で場合分けし，得られる論理関数の値を，2 値の定数接点で表している．また変数の順序を固定し，冗長な接点の削除と等価な接点を共有することで，「既約」な形が一意に得られることが知られている．BDD は，論理関数を比較的少ない記憶量で表現するのに適している．

一方，組合せ集合を表現するには，Zero-suppressed BDD(ZDD)[3] を用いると効率良く扱うことができる．ZDD での簡約化規則は，1-枝が 0-終端接点を直接指している接点を削除する．これによって，組合せ集合に選ばれることのないアイテムに関する接点が自動的に削除されることになり，BDD よりも効率良く表現することができる．ZDD における簡約化規則でも，表現の一意性は保たれる．ZDD を用いると，大規模な頻出パタン集合を圧縮してコンパクトに表現することができ，様々な解析処理が行える．

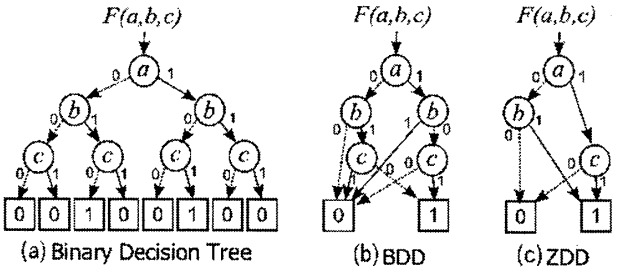

図 2 二分決定木，BDD，ZDD

4. LCM over ZDDs と変数順序付け

4.1 LCM over ZDDs

LCM は宇野ら [4] によって開発された，出力サイズに対して線形時間で頻出アイテム集合を列挙する非常に高速なアルゴリズムである．LCM は深さ優先探索に基づくアルゴリズムの 1 つで，与えられた頻出パタンにアイテムを 1 個ずつ追加しながら再帰的な処理を繰り返すことにより全ての頻出パタンを列挙する．あらかじめ変数順序を定め，その順序に従って再帰的にアイテムを抽出する LCM over ZDDs と呼ぶアルゴリズムが提案されている．膨大な個数の解が生成される場合でも，ZDD により

コンパクトに圧縮して出力できることが特長である．

4.2 LCM over ZDDs での変数順序付け

LCM では低頻度アイテムを上位に置く順序付けの方が高速に動作することが知られている．LCM over ZDDs でもそれに準ずる性質があると思われるが，LCM の再帰呼び出し回数だけでなく，生成される ZDD のサイズも考慮する必要がある．

4.3 動的重み付け法

データマイニング向けの変数順序付け法として，動的重み付け法 [2] と呼ばれる手法が最近提案されている．データベースに含まれるタプルをノードとして補グラフのような構造をつくり，それを辿って変数へ重み付けし，重みの大きい順に上位に置くものである．この結果，低頻度アイテムが上位にくる傾向はあるが，局所計算性のある変数同士が近い順序になり，ZDD 上でのサブグラフが多く共有できるようになる．なお，順序付けにかかる時間は低頻度上位順の方が約 n 倍短い．

5. データベース例題における変数順序付けの影響

5.1 実験

指定した変数順序を入力に ZDD を生成するプログラムを用い，それぞれの変数順序付けを用いて頻出パタンを ZDD で表す．このとき，ZDD のサイズや再帰呼び出し回数にどれだけ差が生まれるか実験を行い確認した．まず最初に適当な閾値を与え，徐々にそれを増やし，サイズと再帰回数の変化を調べた．実験には表 1 に示すような，アイテム数，タプル数，アイテムの平均出現頻度が異なる 11 の例題を使用した．また表 1 では，それぞれの変数順序で生成された ZDD のうち，サイズに最も大きな差が見られたときの閾値とそれぞれのサイズ，再帰回数，サイズ比も示している．

ZDD を生成する際の再帰回数は，どの例題においても大きな差は生まれなかった．つまり低頻度上位順と動的重み付け法を比較すると，計算時間が劇的に変わるということはほとんどないと考えられる．一方生成される ZDD のサイズは，例題によっては顕著な違いが現れた．特に pumsb-star では動的重み付け法の ZDD が，低頻度上位順の約半分のサイズになった．ごく僅かな違いしか見られない例題が多いが，pumsb と pumsb-star のような似た例題でもサイズの大小が逆になる例もあった．

図 3 は pumsb-star の，閾値による ZDD サイズの変化を実験結果からグラフ化したものである．実線は低頻度上位順での ZDD サイズを，破線は動的重み付け法での ZDD サイズを示している．閾値が高くなるにつれ，サイズ差はほぼなくなる．

5.2 LCM over ZDDs 特有の順序付けの影響が表れる例

次に表 2 のような，変数順序によって大きな影響が表れる例題を人工的に作った．a_1 と b_1 を抜いた組合せ，a_2 と b_2 を抜いた組合せ $\cdots$ というように，それぞれ 1 組ず

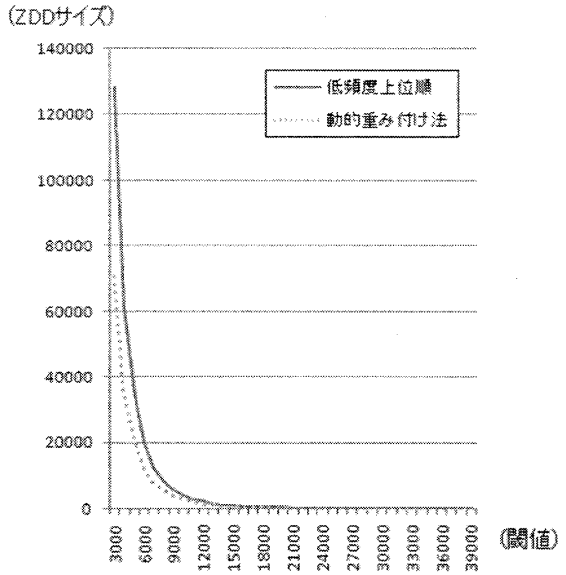

図 3 pumsb-star における ZDD のサイズ変化

ID	レコード
1	$a_2\ a_3\ a_4\ a_5\quad b_2\ b_3\ b_4\ b_5$
2	$a_1\quad a_3\ a_4\ a_5\ b_1\quad b_3\ b_4\ b_5$
3	$a_1\ a_2\quad a_4\ a_5\ b_1\ b_2\quad b_4\ b_5$
4	$a_1\ a_2\ a_3\quad a_5\ b_1\ b_2\ b_3\quad b_5$
5	$a_1\ a_2\ a_3\ a_4\quad b_1\ b_2\ b_3\ b_4$
6	$a_1\ a_2\ a_3\ a_4\ a_5$

表 2 人工的な例題

つ取り除いている．さらに a_1〜a_5 の組合せを最後に追加することで，b_x よりも a_x の頻度を 1 だけ大きくしている．この例題を用いてそれぞれの変数順序を導き出すと，表 3 のようになった．低頻度上位順は頻度にしか着目していないせいで，上位に b が，下位に a がおかれている．一方動的重み付け法では，a_x と b_x が近い位置に並んでいる．パタンの最小出現頻度である閾値 α を増加させながらこれらの変数順序による ZDD サイズを比較した結果を表 4 に示す．特に閾値が低い場合，動的重み付け法の ZDD サイズの方が極めて小さかった．今回は a_1〜a_5, b_1〜b_5 の 10 変数の例題で実験したが，変数の個数を増やせばさらに大きな差が出ると考えられる．

	上位 ⟵ ⟶ 下位
低頻度上位順	$b_2\ b_5\ b_4\ b_1\ b_3\ a_5\ a_4\ a_3\ a_2\ a_1$
動的重み付け法	$b_2\ a_2\ a_3\ b_4\ a_4\ b_5\ a_5\ b_1\ b_3\ a_1$

表 3 例題から求めた変数順序

データベース名	アイテム数	タプル数	平均出現頻度 (%)	閾値	低頻度上位順		動的重み付け法		サイズ比 (%)
					サイズ	再帰回数	サイズ	再帰回数	
chess	75	3196	49.33	900	79652	7468554	78292	7492083	-1.71
mushroom	119	8124	19.32	550	1827	19558	1560	19616	-14.61
BMS-POS	1657	515597	0.39	2000	1207	7052	1190	7052	-1.41
BMS-WV-1	497	59602	0.51	30	110011	6578114	106920	6429932	-2.81
BMS-WV-2	3340	77512	0.14	35	10834	113097	10212	111296	-5.74
accidents	468	340183	7.22	9000	2141	269775	2074	269775	-3.13
T10I4D100K	870	100000	1.16	90	9198	29565	9208	29565	+0.11
T40I10D100K	942	100000	4.20	100	1132008	30232060	1140976	30255839	+0.79
connect	129	67557	33.33	3000	394838	34390408	394587	34390408	-0.06
pumsb	2113	49046	3.50	21000	149999	98055157	191663	98048575	+27.78
pumsb-star	2088	49046	2.41	3000	128263	27285633	70625	26306252	-44.94

(Pentium4, 3.0GHz, Ubuntu 8.04, 主記憶 1GByte)

表 1　実験に用いた例題の特徴と結果

閾値	低頻度上位順	動的重み付け法	サイズ比 (%)
1	59	25	-57.63
2	54	29	-46.30
3	38	26	-31.58
4	18	16	-11.11

表 4　それぞれの変数順序付けによるサイズの違い

6. まとめ

LCM では低頻度上位順で良かったが, LCM over ZDDs では局所計算性の影響で，単純な低頻度上位順ではうまくいかない場合があることがわかった．しかし現実の例題ではほとんどの場合，LCM over ZDDs でも動的重み付け法と低頻度上位順で差異はないので，高速な低頻度上位順で十分なことが多い．アイテムが密な場合には動的重み付け法も試す価値があると思われる．

参考文献

[1] Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation" IEEE Transactions on Computers, Vol.C-35, No.8, pp.677-691, 1986.

[2] 岩崎　玄弥, 湊　真一, トーマス　ツォイクマン, "頻出パター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とその評価" 電子情報通信学会論文誌, Vol.J91-D, No.3, pp.608-618, 2008.

[3] S. Minato, T. Uno and H. Arimura, "LCM over ZBDDs: Fast Generation of Very Large-Scale Frequent Itemsets Using a Compact Graph-Based Representation," in Advances in Knowledge Discovery and Data Mining, pp.234-246, 2008.

[4] T. Uno, M. Kiyomi and H. Arimura, "LCM ver.2: ecient mining algorithms for frequent/closed/maximal itemsets," In Proc. of IEEE ICDM ' 04 Workshop FIMI ' 04, 2004.