TCS-TR-A-07-27

TCS Technical Report

A Theoretical Study on Variable Ordering of ZBDDs for Representing Frequent Itemsets

by

Shin-ichi Minato

Division of Computer Science

Report Series A

May 18, 2007

Hokkaido University
Graduate School of
Information Science and Technology

Email: minato@ist.hokudai.ac.jp

Phone: +81-011-706-7682

Fax: +81-011-706-7682

A Theoretical Study on Variable Ordering of ZBDDs for Representing Frequent Itemsets

Shin-ichi Minato
Division of Computer Science
Hokkaido University
North 14, West 9
Sapporo 060-0814, Japan

May 18, 2007

(Abstract) Recently, an efficient method of database analysis using Zero-suppressed Binary Decision Diagrams (ZBDDs) has been proposed. BDDs are a graph-based representation of Boolean functions, now widely used in system design and verification. Here we focus on ZBDDs, a special type of BDDs, which are suitable for handling large-scale combinatorial itemsets in frequent itemset mining. In general, it is well-known that the size of ZBDDs greatly depends on variable ordering; however, in the specific cases of applying ZBDDs to data mining, the effect of variable ordering has not been studied well. In this paper, we present a theoretical study on ZBDD variable ordering for representing frequent itemsets. We show two typical databases we found out, where the ZBDD sizes are exponentially sensitive to the variable ordering. We also show that there is a case where the ZBDD size must be exponential in any variable ordering. Our theoretical results are helpful for developing a good heuristic method of variable ordering.

1 Introduction

Discovering useful knowledge from large-scale databases has attracted a considerable attention during the last decade. Frequent pattern mining is one of the fundamental problems for knowledge discovery. Since the pioneering paper by Agrawal *et al.*[1], various algorithms have been proposed to solve the frequent pattern mining problem (cf., e.g., [11, 5]).

Recently, we have attacked the problem of efficiently generating the frequent patterns in a transaction database by using a data structure called *Zero-suppressed Binary Decision Diagrams* (abbr. ZBDDs), see [7, 8]. ZBDDs are a special case of Binary Decision Diagrams (abbr. BDDs)[2]. Using ZBDDs one can implicitly enumerate sets of combinations. Moreover, one can then perform efficiently various operations including the discovery and analysis of frequent patterns.

In general, it is well-known that the size of ZBDDs greatly depends on variable ordering, however, in the specific cases of applying ZBDDs to data mining, the effect of variable ordering has not been studied well. In this paper, we present a theoretical study on ZBDD variable ordering for representing frequent itemsets. We show two typical databases we found out, where the ZBDD sizes are exponentially sensitive to the variable ordering. We also show that there is a case where the ZBDD size must be exponential in any variable ordering.

2 Database Representation Using ZBDDs

In this section, we first describe the database representation to be discussed. Here we consider databases of the following type. Let $M \neq \emptyset$ be any set. We refer to the elements of M as to *items*. In our examples below, we use $M = \{a, b, c\}$. Then the set of all possible combinations is the power set $\wp(M)$ of M. Any subset $\mathcal{C} \subseteq \wp(M)$ is said to

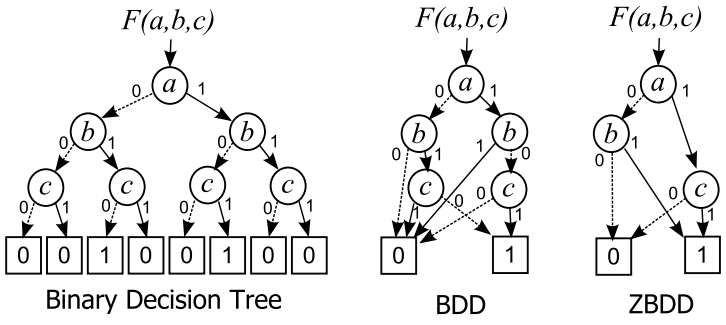

Figure 1: Binary Decision Tree, BDDs and ZBDDs

be a set of combinations. The elements of a set of combinations are sets of items, e.g., $\{a, c\}$. To simplify notation, we write ac instead of $\{a, c\}$ and we refer to the elements of a set of combinations as to *tuples*. A transaction database is just a list of tuples.

2.1 BDDs and ZBDDs

A *Binary Decision Diagram (BDD)* is a graph representation for a Boolean function. An Example is shown in Fig. 1 for $F(a, b, c) = a\overline{b}c \vee \overline{a}b\overline{c}$.

Given a variable ordering (in our example a, b, c), one can use Bryant's algorithm[2] to construct the BDD for any given Boolean function. For many Boolean functions appearing in practice this algorithm is quite efficient and the resulting BDDs are much more efficient representations than binary decision trees.

BDDs were originally invented to represent Boolean functions. But we can also map a set of combinations into Boolean space of n variables, where n is the cardinality of the item set M (see Fig. 2). So, one could also use BDDs to represent sets of combinations. However, one can even obtain a more efficient representation by using *Zero-suppressed* BDDs (ZBDDs)[7].

If there are many similar combinations then the subgraphs are shared resulting in a smaller representation. In addition, ZBDDs have a special type of node deletion rule. As shown in Fig. 3, All nodes whose 1-edge directly points to the 0-terminal node are deleted. Because of this, the nodes of items that do not appear in any sets of combinations are automatically deleted as shown in Fig.1. This ZBDD reduction rule is extremely effective if we handle a set of sparse combinations. If the average appearance ratio of each item is 1%, ZBDDs are possibly more compact than ordinary BDDs, up to 100 times.

ZBDD representation has another good property that each path from the root node to the 1-terminal node corresponds to each combination in the set. Namely, the number of such paths in the ZBDD equals to the number of combinations in the set. This beautiful property indicates that, even if there are no equivalent nodes to be shared, the ZBDD structure explicitly stores all items of each combination, as well as using an explicit linear linked list data structure. In other words, (the order of) ZBDD size never exceeds the explicit representation. If more nodes are shared, the ZBDD is more compact than linear list.

2.2 ZBDD-based representation for frequent itemsets

Frequent itemset mining (or frequent pattern mining) is the problem to enumerate all possible subsets of itemset M (also called patterns) which appear more than or equal to α times in the database, for given α. Since their introduction by Agrawal et al.[1], many papers have been published about new algorithms and improvements for solving such mining problems[4, 5, 11]. Recently, graph-based methods, such as FP-growth[5], have received a great deal of attention, because they can quickly manipulate large-scale itemset data by constructing compact graph structures in main memory.

a	b	c	F	
0	0	0	0	$\rightarrow S$
0	0	1	0	
0	1	0	1	$\rightarrow b$
0	1	1	0	
1	0	0	0	
1	0	1	1	$\rightarrow ac$
1	1	0	0	
1	1	1	0	

As a Boolean function:
$F = a\overline{b}c \vee \overline{a}b\overline{c}$

As a set of combinations:
$S = \{ac,\ b\}$

Figure 2: Correspondence of Boolean functions and sets of combinations.

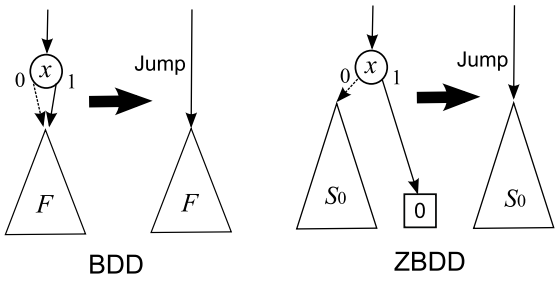

Figure 3: BDD and ZBDD reduction rule.

The ZBDD-based method is a similar approach to handling sets of combinations in main memory but is more efficient because ZBDD is a kind of DAG for representing itemsets, while FP-growth uses a tree representation for the same objects. In general, DAGs can be more compact than trees.

Recently, our research group has developed an efficient algorithm *ZBDD-growth*[9] to generate ZBDDs compactly representing all frequent itemsets for given databases. Our method is not only enumerating/listing the frequent patterns but also efficiently analyzing the huge size of mining results by using ZBDD operations. For example, extracting all patterns including a certain items, or computing the intersection/union/difference set for given two sets of patterns. The computation time of those operations does not directly depend on the number of patterns but almost linear to the (compressed) ZBDD size. It is an important advantage of using ZBDDs.

3 Variable Ordering of ZBDDs for Representing Frequent Itemsets

As described above, it is possible for us to represent histograms of frequent pattern sets compactly by using the ZBDD data structure. However, the ZBDD size is quite sensitive with respect to the underlying ordering of the variables. So, it is important to find a good variable ordering such that the resulting ZBDD size is close to the smallest size possible.

3.1 Properties of Variable Ordering in Ordinary BDDs

In the field of logic VLSI circuit design, many researchers have dealt with the problem of finding good variable orderings for BDDs. For ordinary BDDs, two features of the variable ordering are known that affect the size of the resulting BDDs [3].

(1) Pairs of inputs having the local computability property had better be kept close to one

another in the ordering.

(2) Inputs having a strong controllability to the output had better be located at higher order.

As a typical example where the local computability dominates the BDD size, the following AND-OR two-level logic function is known.

$$x_1x_2 \vee x_3x_4 \vee x_5x_6 \vee \cdots x_{2n-1}x_{2n}$$

This function can be represented by only $2n$ BDD nodes with the variable order: $x_1, x_2, x_3, x_4, \ldots, x_{2n}$, where each pair of variables in the same product term are kept together, while we need $(2^{n+1}-2)$ BDD nodes with the order: $x_1, x_3, \ldots, x_{2n-1}, x_2, x_4, \ldots, x_{2n}$, where those pairs are kept away from each other. (see Fig. 4.)

On the other hand, the data-selector function is known as another example where the output controllability dominates the BDD size. For example, 8-bit data-selector with three control inputs and eight data inputs has the function that one of the data input is just selected by the 3-bit binary code of control inputs. In this case, the function can be represented by a linear size of BDDs when the control inputs are higher than data inputs, but the BDD becomes an exponential size in the reversal order. (see Fig. 5.)

As shown the above, each of the both properties may have an exponential impact to the BDD size. Although one should try to come up with orderings obeying these two features, it may be difficult to do so, since these requirements are sometimes contradictory. The problem of finding the optimal variable ordering for BDDs is known to be NP-complete [10]. In addition, there exist the functions which always require an exponential number of BDD nodes for any variable ordering, so in such cases, it is no use to make ordering.

3.2 Consideration on ZBDDs Representing Frequent Patterns

Now we consider the effect of variable ordering for the ZBDDs representing frequent itemsets. For the sake of simple discussion, first we assume the minimum frequency $alpha = 1$, namely, we consider the ZBDD enumerating all possible patterns which appear at least once in the database. In this case, each tuple with k items generates 2^k patterns, and the total number of patterns may become $O(2^n)$ when n is the size of database description. Thus, the ZBDD size may become exponential in the worst case.

The ZBDD of all patterns in the given database can be generated by computing the union of $P(T_k)$ for all tuples $T_1, T_2, \ldots, T_m$ in the database, where $P(T_k)$ means the sets of all patterns included in a tuple T_k. Here we can observe the following property. If an item x appears in a tuple T_k, the patterns in $P(T_k)$ may or may not include x, therefore, the item x contributes to double the number of patterns in $P(T_k)$. On the other hand, if x does not appear in T_k, the patterns in $P(T_k)$ never include x, and thus the item x does not contribute to increase the patterns.

Let us consider the analogy with the process of generating an ordinary BDD for a given logic expression. When an input variable x appears in a product term T_k in the logic expression, the variable x should be true to satisfy T_k, so the variable x does not contribute to increase the satisfiable solutions for the logic of T_k. On the other hand, if x does not appear in T_k, the value of x may be false or true, both possible to satisfy T_k, so the number of solutions becomes twice.

Consequently, we can observe the completely opposite effects between the two facts that a variable appears in a term of the logic expression, and that an item appears in a tuple of the database. This observation indicates that we may discuss the effect of ZBDD variable ordering by looking the missing items in a tuple as well as the variables appearing in the logic expressions.

3.3 A Typical Example Dominated by Local Computability

Based on the above consideration, we made an artificial database where the property of local com-

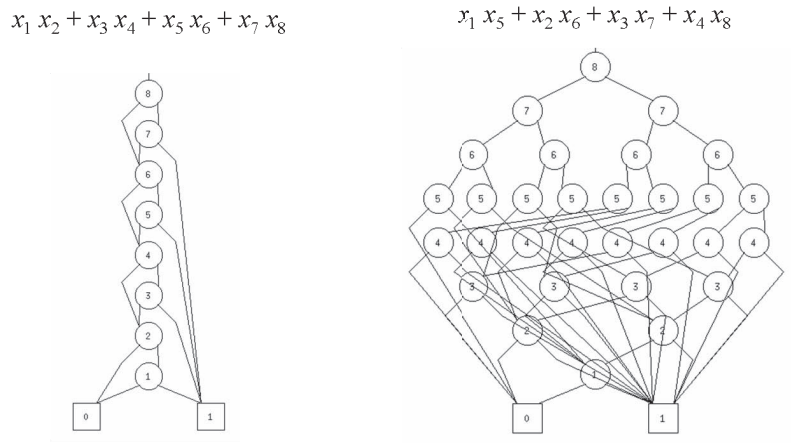

Figure 4: Effect of BDD variable ordering for the AND-OR two-level logic function.

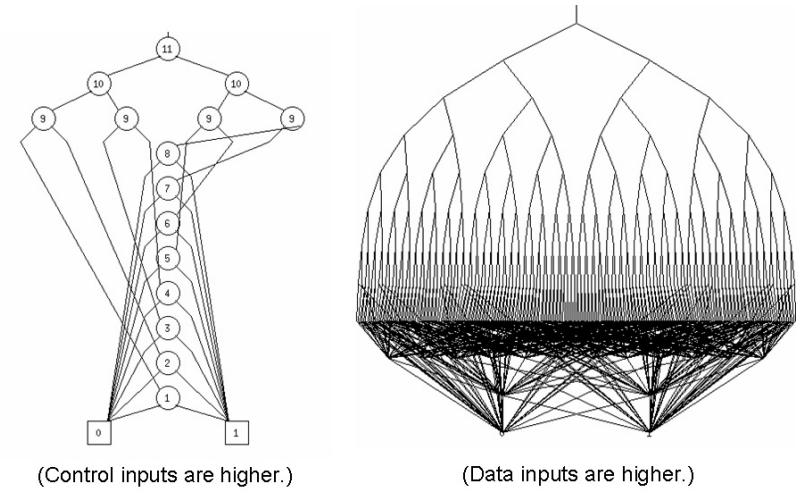

Figure 5: Effect of BDD variable ordering for the data-selector function.

putability dominates the ZBDD size, as follows.

T_1	a_2	a_3	a_4	$\cdots$	a_n	b_2	b_3	b_4	$\cdots$	b_n
T_2	a_1	a_3	a_4	$\cdots$	a_n	b_1	b_3	b_4	$\cdots$	b_n
T_3	a_1	a_2	a_4	$\cdots$	a_n	b_1	b_2	b_4	$\cdots$	b_n
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
T_n	a_1	a_2	a_3	$\cdots$	a_{n-1}	b_1	b_2	b_3	$\cdots$	b_{n-1}

This database consists of the tuples T_k $(k = 1, ..., n)$ each of which has almost all items $a_1\ a_2\ ...\ a_n\ b_1\ b_2\ ...\ b_n$ but only one pair of items a_k and b_k are missing. If we consider the opposite property of item appearance, this database corresponds to the AND-OR two-level logic expression shown in Section 3.1, and we can expect that the pairs of two items missing from the same tuple have the property of local computability.

To confirm our consideration, we generated the ZBDDs representing all patterns included in the database. Table 1 shows the ZBDD size of the two different variable order. The result obviously shows an exponential difference between the two ordering. It is intuitively explained as follows. In the order2, the items a_1 to a_k are ordered in the higher positions, but there is no opportunity to share the ZBDD nodes only by the item a's information without b's, so just a binary tree are generated for the first n stages and thus at least $(2^n - 1)$ nodes are generated. On the other hand, when we use the

Table 1: Experimental results for the databases dominated by local computability.
Order1: $a_1\ b_1\ a_2\ b_2\ \ldots\ a_n\ b_n$
Order2: $a_1\ a_2\ \ldots\ a_n\ b_1\ b_2\ \ldots\ b_n$

n	ZBDD size (order1)	ZBDD size (order2)	Total patterns
3	10	16	37
4	15	39	175
5	20	86	781
6	25	181	3,367
7	25	372	14,197
8	30	755	58,975
9	35	1,522	242,461
10	40	3,057	989,527
11	45	6,128	4,017,157
12	50	12,271	16,245,775

other order such that a_k and b_k are kept closer, the ZBDD nodes can be shared by using combinatorial information of a_k and b_k on each stage, thus the ZBDD size becomes $O(n)$.

3.4 A Typical Example Dominated by Output Controllability

Based on the above consideration, we made an artificial database where the property of output controllability dominates the ZBDD size. The following database corresponds to the 8-bit data-selector function, shown in Section 3.1.

T_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y_0	y_2	y_4
T_1	x_0	x_2	x_3	x_4	x_5	x_6	x_7	y_0	y_2	y_5
T_2	x_0	x_1	x_3	x_4	x_5	x_6	x_7	y_0	y_3	y_4
T_3	x_0	x_1	x_2	x_4	x_5	x_6	x_7	y_0	y_3	y_5
T_4	x_0	x_1	x_2	x_3	x_5	x_6	x_7	y_1	y_2	y_4
T_5	x_0	x_1	x_2	x_3	x_4	x_6	x_7	y_1	y_2	y_5
T_6	x_0	x_1	x_2	x_3	x_4	x_5	x_7	y_1	y_3	y_4
T_7	x_0	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_3	y_5

This database has the items $x_0\ x_1\ \ldots\ x_{n-1}$ as the data inputs and $y_0\ y_1\ \ldots\ y_m$ $(m = 2\lceil \log_2 n \rceil)$ as the control inputs. The control inputs have the pair-wise structure as $(y_0, y_1), (y_2, y_3), \ldots$, each of which represents a digit of binary coded number. Either of odd or even numbered y appears on each tuple T_k in the database, to represent the value 0/1 of the binary code for k. The tuple T_k also includes all items x_0 to x_n except x_k. With the opposite property of item appearance, we can consider that this database selects one of the data input x_k according to the binary coded number specified by the control inputs, as well as the data-selector function shown in Section 3.1.

When generating a ZBDD for the sets of all possible patterns included in this database, we can expect that the ZBDD will become a linear size when the control inputs y's are higher than data inputs x's, but it will become an exponential size in the reversal order. It is explained as follows. If we first assign a set of values into all the items y's, the rest of patterns of x's are related only to one or two tuples in the database, so each ZBDD subgraph for items x's becomes a beautiful array structure with n nodes. A pair of (y_i, y_{i+1}) may cause three patterns: only y_i appears, only y_{i+1} appears, or both absent, so, the upper part of ZBDD for y's become a ternary tree for the $\lceil \log_2 n \rceil$-bit of binary code. The total ZBDD nodes are bounded by $O(n \cdot 3^{\log_2 n}) \approx O(n^{2.7})$.

On the other hand, when we use the reversal order such that x's are higher than y's, there is no opportunity to share the ZBDD nodes only by x's information without y's, so just a binary tree are generated for the first n stages and thus at least $(2^n - 1)$ nodes are required.

To confirm our consideration, we generated the ZBDDs representing all patterns included in the database. Table 2 shows the ZBDD size of the two different variable order. The result obviously shows an exponential difference between the two

Table 2: Experimental results for the databases dominated by output controllability.

n (number of x's)	ZBDD size(y's higher)	ZBDD size(x's higher)	Total patterns
8	126	579	5,023
12	339	5,117	227,295
16	650	137,444	4,159,487
20	1,151	2,435,284	161,496,559

orders. Thus, we can see that there exists an example where the property of output controllability has an exponential impact for the ZBDD size.

3.5 A Case of Generating Exponential ZBDD in Any Variable Ordering

Based on the above observation, we can show that there exists a database where the ZBDD representing the set of patterns must be an exponential size in any variable ordering. The "data-selector" database, shown in the last section, consists of the two sorts of items x's and y's. We must put the y's higher than x's to avoid exponential explosion of ZBDD size. Now, let us define N as the total number of items $(= n + 2\lceil \log_2 n \rceil)$, then we make N copies of the databases with the N different variable orders by rotating the items one by one, as shown in Fig. 6, and finally we merge the all N blocks into one database. If we generate a ZBDD for the sets of all patterns in this database, any variable ordering cannot avoid a bad variable order, because, at least one of the blocks, more than $N/2$ items of x's become higher than y's. So, we need $O(2^{N/2})$ ZBDD nodes for representing the patterns for one of the blocks with a bad variable order. Since the initial data size of this database is $O(N^3)$, we can conclude that this database requires an exponential size of ZBDD in any variable ordering.

3.6 Effect of the Minimum Frequency Threshold

In the above discussions, we assume the minimum frequency threshold $\alpha = 1$, it means that the ZBDDs represent the sets of all possible patterns included in the databases. However, in the real applications, we specify a larger α to reduce the number of frequent patterns into a feasible amount, and the size of the ZBDD is also reduced. If the number of frequent patterns are not exponential for a given large α, then the ZBDD size never become exponential, and in such cases, the variable ordering does not have an exponential effect to the ZBDD size.

Table 3 is the experimental result for the same database as shown in Section3.3 with a different threshold $\alpha = n/2$, namely, representing the set of frequent patterns which appears at more than half tuples in the database. The result shows that the variable ordering still has an exponential impact to the ZBDD size when $\alpha = n/2$. In this example, we expect that as long as we specify an α proportional to n, the similar exponential impact will be observed.

4 Conclusion

In this paper, we presented a theoretical study on ZBDD variable ordering for representing frequent itemsets. We found the two typical databases where the ZBDD sizes are exponentially sensitive to the variable ordering, and we discussed why such a remarkable difference occurs. In addition, we also showed that there is a case where the ZBDD size must be exponential in any variable ordering. These discussions clarify the property of variable ordering when we apply the ZBDD-based data structure to data mining problems. Recently, we proposed a heuristic variable ordering method[6] that finds a good variable order before generating ZBDDs, by using the structural information of the given database. Such a heuristic method is developed based on the theoretical results discussed in this paper.

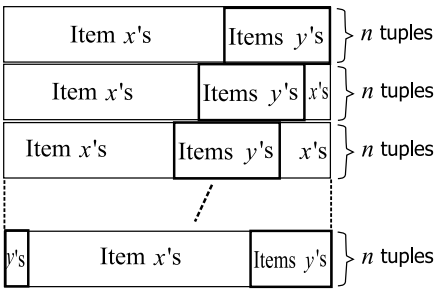

Figure 6: N copies of "data-selector" databases with the rotated variable orders.

Table 3: Same database as Table 1 with the different threshold $\alpha = n/2$.
Order1: $a_1\ b_1\ a_2\ b_2\ \ldots\ a_n\ b_n$
Order2: $a_1\ a_2\ \ldots\ a_n\ b_1\ b_2\ \ldots\ b_n$

n	ZBDD size (order1)	ZBDD size (order2)	Total patterns
3	8	8	67
4	16	30	106
5	22	47	781
6	33	142	694
7	42	222	1,156
8	56	616	7,459
9	68	969	12,896
10	85	2,564	81,922
11	100	4,074	143,980
12	120	10,503	912,718

Acknowledgment

The author would like to thank Thomas Zeugmann and Hiroki Arimura of Hokkaido University for their valuable discussions and comments. This research was partially supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (B) on "ZBDD-based Database Analysis," 17300041, 2007, and Grant-in-Aid for Specially Promoted Research on "Semi-Structured Data Mining," 17002008.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data, Vol. 22(2) of SIGMOD Record*, pages 207–216, 1993.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

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

[4] B. Goethals. Survey on frequent pattern mining, 2003. `http://www.cs.helsinki.fi/u/goethals/publications/survey.ps`.

[5] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[6] H. Iwasaki, S. Minato, and T. Zeugmann. A method of variable ordering

for zero-suppressed binary decision diagrams in data mining applications. In *Proc. of The Third IEEE International Workshop on Databases for Next-Generation Researchers (SWOD2007), collocated with ICDE-2007*, 2007.

[7] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proc. of 30th ACM/IEEE Design Automation Conference*, pages 272–277, 1993.

[8] S. Minato. Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer (STTT), Springer*, 3(2):156–170, 2001.

[9] S. Minato and H. Arimura. Frequent pattern mining and knowledge indexing based on zero-suppressed BDDs. In *The 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID'06)*, pages 83–94, 9 2006.

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

[11] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(2):372–390, 2000.