TCS-TR-A-06-14

TCS Technical Report

Symmetric Item Set Mining Using Zero-suppressed BDDs

by

Shin-ichi Minato

Division of Computer Science

Report Series A

May 25, 2006

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

Symmetric Item Set Mining Using Zero-suppressed BDDs

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

May 25, 2006

(Abstract) In this paper, we propose a method for discovering hidden information from large-scale item set data based on the symmetry of items. Symmetry is a fundamental concept in the theory of Boolean functions, and there have been developed fast symmetry checking methods based on BDDs (Binary Decision Diagrams). Here we discuss the property of symmetric items in data mining problems, and describe an efficient algorithm based on ZBDDs (Zero-suppressed BDDs). The experimental results show that our ZBDD-based symmetry checking method is efficiently applicable to the practical size of benchmark databases.

1 Introduction

Frequent item set mining is one of the fundamental techniques for knowledge discovery. Since the introduction by Agrawal et al.[1], the frequent item set mining and association rule analysis have been received much attentions from many researchers, and a number of papers have been published about the new algorithms or improvements for solving such mining problems[3].

After generating frequent item set data, we sometimes faced with the problem that the results of item sets are too large and complicated to retrieve useful information. Therefore, it is important for practical data mining to extract the key structures from the item set data. Closed/maximal item set mining[10] is one of the useful method to find important item sets. Disjoint decomposition of item set data[8] is another powerful method for extracting hidden structures from frequent item sets.

In this paper, we propose one more interesting method for finding hidden structure from large-scale item set data. Our method is based on the symmetry of items. It means that the exchange of a pair of symmetric items has completely no effect for the database information. This is a very strict property and it will be a useful association rule for the database analysis.

The symmetry of variables is a fundamental concept in the theory of Boolean functions, and the method of symmetry checking has been studied for long time in VLSI logic design area. There are some state-of-the-art algorithms[9, 5] using BDDs (Binary Decision Diagrams)[2] to solve such a problem. The BDD-based techniques can be applied to data mining area. Recently, we found that ZBDDs (Zero-suppressed BDDs)[6] are very suitable for representing large-scale item set data used in transaction database analysis[7].

In this paper, we discuss the property of symmetric items in transaction database, and then present an efficient algorithm to find all symmetric item sets using ZBDDs. We also show some experimental results for conventional benchmark data. Our method will be useful for extracting hidden information from a given database.

The paper is organized as follows: First, we review the concept for symmetric variables for Boolean functions in Section 2. In Section 3, we discuss the property of symmetric items in data mining, and describe the algorithm for finding symmetric item sets. We then show the experimental results in section 4, followed by conclusion.

2 Symmetry of variables in Boolean functions

The symmetry is a fundamental concept in the theory of Boolean functions. A symmetric Boolean function means that any exchange of input variables has no effect for the output value. In other words, the output value is decided only by the total number of true assignments in the n-input variables. The parity check functions and threshold functions are typical examples of symmetric functions.

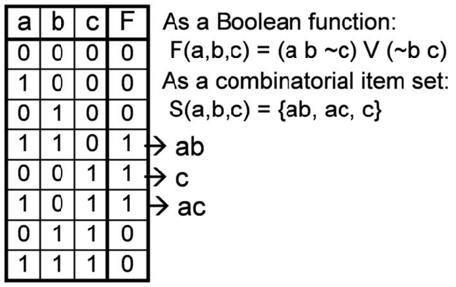
When the function is not completely symmetric, we sometimes find partial groups of symmetric variables. If a pair of variables are exchangeable without any output change, we call them symmetric variables in the function. An obvious property holds that if the pairs (a, b) and (a, c) are both symmetric, then any pair of (a, b, c) is symmetric.

As finding symmetric variables leads to compact logic circuits, it has been studied for long time in VLSI logic design area. In order to check the symmetry of the two variables v_1 and v_2 in the function F, we first extract four sub-functions: F_{00}, F_{01}, F_{10}, and F_{11} by assigning all combinations of constant values 0/1 into v_1 and v_2, and then compare of F_{01} and F_{10}. If the two are equivalent, we can see the two variables are symmetric. In principle, we need $n(n-1)/2$ times of symmetry checks for all possible variable pairs. There have been proposed some state-of-the-art algorithms[9, 5] using BDDs (Binary Decision Diagrams)[2] to solve such a problem efficiently.

3 Symmetric item sets in transaction databases

3.1 Combinatorial item sets and Boolean functions

A combinatorial item set consists of the elements each of which is a combination of a number of items. There are 2^n combinations chosen from n items, so we have 2^{2^n} variations of combinatorial item sets. For example, for a domain of five items



a	b	c	F	
0	0	0	0	
1	0	0	0	
0	1	0	0	
1	1	0	1	→ ab
0	0	1	1	→ c
1	0	1	1	→ ac
0	1	1	0	
1	1	1	0	

As a Boolean function:
F(a,b,c) = (a b ~c) V (~b c)
As a combinatorial item set:
S(a,b,c) = {ab, ac, c}

Figure 1: A Boolean function and a combinatorial item set.

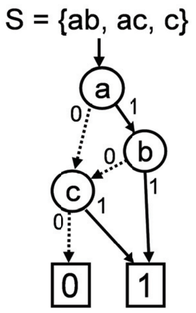


Figure 2: An example of ZBDD.

a, b, c, d, and e, we can show examples of combinatorial item sets as:
$\{ab, e\}$, $\{abc, cde, bd, acde, e\}$, $\{1, cd\}$, 0. Here "1" denotes a combination of null items, and 0 means an empty set. Combinatorial item sets are one of the basic data structure for various problems in computer science, including data mining.

A combinatorial item set can be mapped into Boolean space of n input variables. For example, Fig. 1 shows a truth table of Boolean function: $F = (ab\overline{c}) \vee (\overline{b}c)$, but also represents a combinatorial item set $S = \{ab, ac, c\}$. Using BDDs for the corresponding Boolean functions, we can implicitly represent and manipulate combinatorial item set. In addition, we can enjoy more efficient manipulation using "Zero-suppressed BDDs" (ZBDD)[6], which are special type of BDDs optimized for handling combinatorial item sets. An example of ZBDD is shown in Fig. 2.

The detailed techniques of ZBDD manipulation are described in the articles[6]. A typical ZBDD package supports cofactoring operations to traverse 0-edge or 1-edge, and binary operations between two combinatorial item sets, such as union, intersection, and difference. The computation time for each operation is almost linear to the number of ZBDD nodes related to the operation.

3.2 Symmetric Items in combinatorial item sets

Here we discuss the symmetry of items in a combinatorial item set. For example we consider the following combinatorial item set: $S = \{abc, acd, ad, bcd, bd, c, cd\}$. In this case, the item a and b are symmetric but the other pairs of variables are not symmetric. The symmetry can be confirmed as follows. First we classify the combinations into four categories: (1) both a and b included, (2) only a is included, (3) only b is included, and (4) neither included. Namely, it can be written as: $S = abS_{11} \cup aS_{10} \cup bS_{01} \cup S_{00}$. Then, we can determine the symmetry of a and b by comparing S_{10} and S_{01}. If the two subsets are equivalent, a and b are exchangeable. For the above example, $S_{11} = \{c\}, S_{10} = \{cd, d\}, S_{01} = \{cd, d\}$, and $S_{00} = \{c, cd\}$. We can see a and b are symmetric as $S_{10} = S_{01}$.

Even if we do not know the actual meaning of the item a and b in the original database, we can expect that a and b would have somehow strong relationship if the symmetric property holds. It is a kind of hidden information. It would be a useful and interesting task to find all possible symmetric item sets from the given databases. This method can be used not only for original database but also for frequent item set data to find some relationships between the items.

3.3 ZBDD-based algorithm for finding symmetric item sets

As shown in article[7], the ZBDD-based data structure is quite effective (exponentially in extreme cases) for handling transaction databases, especially when the item sets include many similar partial combinations. Now we show an efficient algorithm of finding symmetric item sets based on ZBDD operations.

First we explain the *cofactor* operation on ZBDDs. Cofactor(S, v) classifies a combinatorial item set S into the two subsets, one of which includes the item v and the other does not. Namely, it extracts S_1 and S_0 such that $S = vS_1 \cup S_0$. If the item v is the top (highest ordered) item in the ZBDD, then S_1 and S_0 are the two sub-graphs pointed by 1-edge and 0-edge of the top decision node, and the cofactor operation can be done in a constant time. Therefore, if the item v_1 and v_2 are the first and second top items in the ZBDD, the symmetry checking is quite easy because S_{10} (subset with v_1 but not v_2) and S_{01} (subset with v_2 but not v_1) can be extracted and compared in a constant time.

When v_1 and v_2 are not in the highest oreder in the ZBDD, we may use recursive expansion for the symmetry checking. We get the top item t in the ZBDD S, and extract S_1 and S_0 as the cofactors of S by t. We then recursively check the symmetry of (v_1, v_2) for each subset S_1 and S_0, and if they are symmetric for the both, we can see they are symmetric for S.

This procedure may require an exponential number of recursive calls in terms of the number of items higher than v_1, v_2 in the ZBDD, however, we do not have to execute the procedure twice for the same ZBDD node because the results will be the same. Therefore, the number of recursive calls is bounded by the ZBDD size, by using a hash-based cache to save the result of procedure for each ZBDD node. In addition, if we found the two items are asymmetric either for S_1 or S_0, we may immediately quit the procedure and conclude they are asymmetric for S. The detailed algorithm is shown with a pseudo code in Fig. 3. Repeating this procedure for all item pairs in S, we can extract all possible symmetric item sets. The total computation time is $O(n^2|G|)$, where n is number of items and $|G|$ is the ZBDD size for S. The time will be shorter in practice, when the most of item pairs are asymmetric, and/or the hash-based cache hits very well in repeating the procedure.

4 Experimental Results

We implemented our symmetric checking algorithm. The program is based on our own ZBDD package, and additional 70 lines of C++ code for

```
SymChk($S, v_1, v_2$) /* Assume $v_1$ higher than $v_2$ in the ZBDD ordering.*/
{
  if ($S = 0$ or $S = 1$) return 1 ;
  $r \leftarrow$ Cache($S, v_1, v_2$) ;
  if ($r$ exists) return $r$ ;
  $t_1 \leftarrow S.top$ ; /* Top item in $S$ */
  if ($t_1$ higher than $v_1$)
    ($S_1, S_0$) $\leftarrow$ (Cofactors of $S$ by $t_1$) ;
    $r \leftarrow$ SymChk($S_1, v_1, v_2$) && SymChk($S_0, v_1, v_2$) ;
  else
    ($S_1, S_0$) $\leftarrow$ (Cofactors of $S$ by $v_1$) ;
    $t_2 \leftarrow$ Max($S_1.top, S_0.top$) ; /* Top item in $S_1, S_0$ */
    if ($t_2$ higher than $v_2$)
      ($S_{11}, S_{10}$) $\leftarrow$ (Cofactors of $S_1$ by $t_2$) ;
      ($S_{01}, S_{00}$) $\leftarrow$ (Cofactors of $S_0$ by $t_2$) ;
      $r \leftarrow$ SymChk(($t_2S_{11} \cup S_{10}$), $t_2, v_2$) && SymChk(($t_2S_{01} \cup S_{00}$), $t_2, v_2$) ;
    else
      ($S_{11}, S_{10}$) $\leftarrow$ (Cofactors of $S_1$ by $v_2$) ;
      ($S_{01}, S_{00}$) $\leftarrow$ (Cofactors of $S_0$ by $v_2$) ;
      $r \leftarrow$ ($S_{10} = S_{01}$)? 1 : 0 ;
    endif
  endif
  Cache($S, v_1, v_2$) $\leftarrow r$ ;
  return $r$ ;
}
```

Figure 3: Sketch of the symmetry checking algorithm.

Table 1: Experimental result

Data name	#Item	#Record	#Tuple	ZBDD nodes	Time(sec) for ZBDD gen.	Sym. pairs	Time(sec) for sym.chk.
mushroom	119	8,124	8,124	8,006	1.1	19	0.6
T10I4D100K	870	100,000	89,135	547,777	59.2	0	61.7
pumsb	2,113	49,046	48,474	1,749,775	166.7	90	1,152.0
BMS-Web-View-1	497	59,602	18,473	42,629	24.9	6	30.2
accidents	468	340,183	339,898	3,876,468	127.5	11	18.0

the symmetry checking algorithm. We used a Pentium-4 PC, 800MHz, 1GB of main memory, with SuSE Linux 9. We can manipulate up to 20,000,000 nodes of ZBDDs in this PC.

For evaluating the performance, we applied our method to the practical databases chosen from FIMI2003 benchmark set[4]. We first constructed a ZBDD for the set of all tuples in the database, and then apply our symmetry checking algorithm for the ZBDD. The results are shown in Table 1. "Sym. pairs" shows the number of symmetric pairs we found. Our result demonstrates that we succeeded in extracting all symmetric item sets for a practical size of databases within a feasible computation time. We can see that no symmetric pairs are found in "T10I4D100K." It is reasonable because this data is randomly generated and there is no strong relationship between any pair of items.

We also conducted another experiment of symmetry checking for the set of frequent patterns in "mushroom." Figure 4 shows the results with the minimum support = 500 and 200. While the data includes a huge number of patterns, ZBDDs are in a feasible size, and our method quickly found the symmetric item groups (shown by parentheses) in a few seconds of computation time.

Minimum support = 500: (Total patterns: 1,442,504, ZBDD nodes: 4,011)

```
(x32 x7 x31) (x60 x64) x119 x48 x102 x91 x58 x80 x101 x95 x66 x61 x29 x17 x78
x68 x69 x77 x45 x117 x116 x56 x6 x111 x11 x44 x110 x43 x42 x94 x53 x37 x28
x24 x16 x10 x41 x15 x114 x99 x55 x39 x14 x2 x107 x98 x93 x90 x86 x85 x76 x67
x63 x59 x54 x52 x38 x36 x34 x23 x13 x9 x3 x1
```

Minimum support = 200: (Total patterns: 18,094,822, ZBDD nodes: 12,340)

```
(x80 x71 x79 x70) (x32 x31) (x78 x68) (x27 x26) x35 x7 x119 x48 x112 x102 x91
x95 x66 x61 x29 x17 x46 x69 x77 x45 x60 x117 x65 x116 x56 x6 x111 x64 x11
x58 x101 x44 x110 x43 x42 x109 x94 x53 x37 x28 x24 x16 x10 x115 x41 x15 x4
x114 x108 x99 x55 x39 x14 x2 x113 x107 x98 x93 x90 x86 x85 x76 x67 x63 x59
x54 x52 x40 x38 x36 x34 x25 x23 x13 x9 x3 x1
```

Figure 4: Results for the all frequent patterns in “mushroom” data.

5 Conclusion

In this paper, we presented an efficient method for extracting all symmetric item sets in transaction databases. The experimental results show that our method is very powerful and will be useful for discovering hidden interesting information in the given data. Now we are going to apply our method to actual real-life data mining problems.

As our future work, we are considering more efficient algorithm to be applied for more larger ZBDDs, and it would also be interesting to develop “approximately” symmetry checking method which allows some errors or noise in the data.

Acknowledgment

The author thanks Prof. Arimura of Hokkaido Univ. for his technical comments. This study is partly supported by Grant-in-Aid Scientific Research (B), 2005, 17300041.

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, edtors, *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, Vol. 22(2) of SIGMOD Record, pp. 207–216, ACM Press, 1993.

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

[3] B. Goethals, “Survey on Frequent Pattern Mining”, Manuscript, 2003. `http://www.cs.helsinki.fi/u/goethals/publications/survey.ps`

[4] B. Goethals, M. Javeed Zaki (Eds.), Frequent Itemset Mining Dataset Repository, Frequent Itemset Mining Implementations (FIMI’03), 2003.
`http://fimi.cs.helsinki.fi/data/`

[5] N. Kettle and A. King: “An Anytime Symmetry Detection Algorithm for ROBDDs,” In Proc. IEEE/ACM 11th Asia and South Pacific Design Automation Conference (ASPDAC-2006), pp. 243–248, Jan. 2006.

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

[7] S. Minato and H. Arimura: ”Efficient Combinatorial Item Set Analysis Based on Zero-Suppressed BDDs”, IEEE/IEICE/IPSJ International Workshop on Challenges in Web Information Retrieval and Integration (WIRI-2005), pp. 3–10, Apr., 2005.

[8] S. Minato: “Finding Simple Disjoint Decompositions in Frequent Itemset Data Using Zero-suppressed BDD,” In Proc. of IEEE ICDM 2005 workshop on Computational Intelligence in Data Mining, pp. 3-11, ISBN-0-9738918-5-8, Nov. 2005.

[9] A. Mishchenko, “Fast Computation of Symmetries in Boolean Functions,” IEEE Trans.

Computer-Aided Design, Vol. 22, No. 11, pp. 1588–1593, 2003.

[10] T. Uno, T. Asai, Y. Uchida and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases," In Proc. of the 8th International Conference on Discovery Science 2004 (DS-2004), 2004.