アルゴリズム 32－5
（1993. 3. 18）

論理関数による組合せアルゴリズムの解法 II

仙波一郎
茨城大学 教養部

矢島脩三
京都大学 工学部 情報工学科

従来、種々の組合せ問題を解くのにバックトラック技法が使われてきた。しかし、効率よく解くには問題ごとにデータ構造を工夫しなければならない。この論文では、論理関数を用いると多くの組合せ問題が問題に即して素直にかつ簡潔に表現できることを示す。例として、集合分割問題を取り上げ、論理関数を二分決定グラフ（A Binary Decision Diagram, BDD）で記述し、二分決定グラフを高速に効率よく処理するＢＤＤパッケージを使って解く。

Combinatorial Algorithms by Boolean Processing II

Ichiro Semba
College of General Education
Ibaraki University

Shuzo Yajima
Department of Information Science
Faculty of Engineering
Kyoto University

So far, backtrack technique has been used to solve various problems of generating combinatorial objects. However, in order to obtain solutions efficiently, we need to make efforts to find out suitable data structures according to each problem. In this paper, we show many combinatorial problems can be written briefly by using Boolean functions. As an example, we will examine in detail the problem of generating all the partitions of the set {1,2,...,n}. The Boolean functions have been represented by a Binary Decision Diagram (BDD) and manipulated by an efficient BDD manipulator.

1. Introduction

So far, backtrack technique has been used to solve various problems of generating combinatorial objects. However, in order to obtain the satisfying solutions, we need to make efforts to find out suitable data structures according to each combinatorial problem. On the other hand, it is relatively easy to represent many combinatorial problems by using Boolean functions.

Recently, efficient Boolean function manipulators have been developed and widely used[1][2][3]. Thus, in this paper, we will represent various combinatorial problems by using Boolean functions. We have solved these problems efficiently by using BDD manipulator on a Sun SPARC station 2 workstation(64MByte).

The following problems have been considered.
The BDDsize is defined as the number of nodes required to construct the BDD` representing combinatorial objects.

(1) The problem of generating all the r-permutations of the set {1,2,...,n} with unlimited repetitions

An r-permutation of the set {1,2,...,n} (r-permutation of n for short) with unlimited repetitions is defined as an ordered arrangement of r of the set {1,2,...,n} with unlimited repetitions.
Example: all the 2-permutations of 4 with unlimited repetitions are
11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, 41, 42, 43, 44.
Experimental results: n=10, r=100, solutions=10^{100}, BDDsize=1800.

(2) The problem of generating all the r-permutations of n

Example: all the 2-permutations of 4 are
12, 13, 14, 21, 23, 24, 31, 32, 34, 41, 42, 43.
Experimental results: n=10, r=10, solutions=3628800, BDDsize=16398.

(3) The problem of generating all the r-permutations of n with restrictions on absolute positions

The derangement problem has been considered. An n-permutation of n is said to be a derangement of n, if the number i does not appear in ith position for $1 \leqq i \leqq n$.
Example: all the derangements of 4 are
2143, 4123, 2413, 3142, 3412, 4312, 2341, 3421, 4321.
Experimental results: n=10, solutions=1334961, BDDsize=16104.

(4) The problem of generating all the r-permutations of n with restrictions on relative positions

The problem of generating all the stack permutations of n has been considered. A stack permutation of n is defined as an n-permutation of n, $p_1 p_2 \ldots p_n$, such that no subsequence $p_i p_j p_k (1 \leqq i < j < k \leqq n$, $p_j < p_k < p_i)$ exists. The number of all the stack permutations of n is ${}_{2n}C_n/(n+1)$. This number is well known as the Catalan number.
Example: all the stack permutations of 4 are 1234, 2134, 1324, 2314, 3214, 1243, 2143, 1342, 1432, 2341, 3241, 2431, 3421, 4321.
Experimental results: n=10, solutions=16796, BDDsize=10732.

(5) The problem of generating all the r-partitions of the set {1,2,...,n}
See Section 2.

(6) The problem of generating all the r-covers of the set {1,2,...,n}

We denote all subsets of the set {1,2,...,n} by $S_i \neq \phi (1 \leqq i \leqq 2^n-1)$
An r-cover of the set {1,2,...,n} (r-cover of n for short) is defined as a collection of r subsets $S_{i_1}, S_{i_2}, \ldots, S_{i_r} (1 \leqq i_1 < i_2 < \ldots < i_r \leqq 2^n-1)$ such that $\bigcup_{j=1}^{r} S_{i_j} = \{1,2,\ldots,n\}$, $S_{i_j} \not\subset S_{i_k} (i_j \neq i_k)$.

Example: n=3, $S_1=\{1\}, S_2=\{2\}, S_3=\{1,2\}, S_4=\{3\}, S_5=\{1,3\}, S_6=\{2,3\}, S_7=\{1,2,3\}$

All the 1-cover of 3 is $\{S_7\}$.

All the 2-covers of 3 are $\{S_5, S_6\}, \{S_3, S_6\}, \{S_1, S_6\}, \{S_3, S_5\}, \{S_2, S_5\}, \{S_3, S_4\}$.

All the 3-covers of 3 are $\{S_1, S_2, S_4\}, \{S_3, S_5, S_6\}$.

Experimental results: n=6, r=10, solutions=1067759, BDDsize=103417.

(7) The problem of generating all the increasing k-subsequences

We assume that a sequence of n distinct positive integers 1,2,...,n is given. An increasing k-subsequence of a sequence, $a_1 a_2 \ldots a_n$, is defined as an increasing subsequence of a sequence of length k.
Experimental results: n=16, k=8, $a_1 a_2 \ldots a_n = 12 \ldots n$,
solutions=12870, BDDsize=79.

(8) The problem of generating all the k-common substrings

We assume that a string is any finite sequence of elements from an alphabet. Let u and v be a string.

A string w is called an k-common substring(k-CS for short) of u and v, if w is a substring of length k of both u and v. If k is the maximum length, an k-CS is called a longest common subsequence(LCS for short).
Experimental results: u=abab...ab, |u|=12, v=abcabc...abc, |v|=12,
k=8, solutions=45, BDDsize=168.

(9) The problem of generating all the partitions of an integer n

A partition of an integer is a division of the integer into positive integral part, in which the order of these parts is not important.
Example: 5 different partitions for n=4, 4=3+1=2+2=2+1+1=1+1+1+1
Experimental results: n=30, solutions=5604, BDDsize=25418.

(10) The problem of generating all the r-addition chains of n

An addition chain of an integer n of length r (r-addition chain of n for short) is a sequence of r+1 integers $a_0, a_1, \ldots, a_r$ such that (1) $a_0=1, a_1=2, a_r=n$ and (2) for each i, $a_i = a_j + a_k$ $(1 \leqq j \leqq k < i \leqq r)$.
Experimental results: n=36, r=7, solutions=12, BDDsize=88.

We note that the length of an addition chain of n is equal to the number of multiplications required to compute x^n.

(11) The $n \times n$ nonattacking queens problem

This problem is to find all the ways to place n queens on an $n \times n$ chess board so that no two queens are attacking each other.
Experimental results: n=8, solutions=92, BDDsize=2373.

(12) The problem of generating all the $n \times n$ Latin squares

An $n \times n$ Latin square (Latin square of order n) is defined as an $n \times n$ arrangement of the numbers 1,2,3,...,n in such a way that no number appears twice in the same row and in the same column.
Experimental results: n=5, solutions=56, BDDsize=1941.

(13) The problem of generating all the balanced incomplete block designs

Let $A=\{a_1, a_2, \ldots, a_v\}$ be a set of v objects. An k-subset of A is a subset containing k objects of the set A. A balanced incomplete block design of A is defined as a collection of b k-subsets of A (denoted by $B_1, B_2, \ldots, B_b$ and called the blocks).
The b blocks satisfies the following conditions.
(1) Each object appears in exactly r of the b blocks.
(2) Every two objects appear simultaneously in exactly λ of the b blocks.
(3) $k<v$.

Example: b=10, v=6, r=5, k=3, λ=2. One solution is
$B_1=\{a_1, a_2, a_3\}$, $B_2=\{a_1, a_2, a_6\}$, $B_3=\{a_1, a_3, a_4\}$, $B_4=\{a_1, a_3, a_6\}$,
$B_5=\{a_1, a_4, a_5\}$, $B_6=\{a_2, a_3, a_4\}$, $B_7=\{a_2, a_3, a_5\}$, $B_8=\{a_2, a_4, a_6\}$,
$B_9=\{a_3, a_5, a_6\}$, $B_{10}=\{a_4, a_5, a_6\}$.
Experimental results: b=10, v=6, r=5, k=3, λ=2, solutions=12, BDDsize=463.

(14) The problem of finding all the shortest paths

Let N be a set of nodes and E be a set of edges. We assume that an undirected graph G=(N,E) where each edge has a weight 1 is given. This problem is to find the shortest path length from node S (a source node) to node D (a destination node) and obtain the actual paths. The 4-cube has been considered and all paths from node S to D such that Hamming distance between S and D is 4 are examined.
Experimental results: solutions=24, BDDsize=260.

(15) The problem of finding all Hamilton cycles

We assume that an undirected graph G=(N,E) is given. This problem is to find all cycles, if the graph G has a cycle which passes through each node exactly once (for all nodes of the graph G).
The 3-cube has been considered.
Experimental results: solutions=6, BDDsize=378.

(16) The problem of finding all the maximal matchings

A graph G=(N,E) is bipartite if there exists two disjoint subsets X and Y ($X \cup Y=N, X \cap Y=\phi$) such that no node in a subset is adjacent to nodes in the same subset. A matching in a bipartite graph is a subset of E such that no two edges in the subset are incident with the same node. A matching with the maximum number of edges in it is called a

maximal matching.
Experimental results: $X=\{1,2,3,4,5,6\}$, $Y=\{7,8,9,10,11,12\}$, $E=\{(x,y) \mid \text{for any } x \in X, y \in Y\}$, solutions=720, BDDsize=522.

(17) The problem of generating all the minimal dominating sets

We assume that an undirected graph G=(N,E) is given.
A set of nodes in an undirected graph G is called to be a dominating set if every node not in the set is adjacent to one or more nodes in the set. A minimal dominating set is a dominating set such that no proper subset of it is also a dominating set.
The 4-cube has been examined.
Experimental results: solutions=40, BDDsize=174.

(18) The problem of generating all the maximal independent sets

We assume that an undirected graph G=(N,E) is given.
A set of nodes in an undirected graph G is called to be an independent set of no two nodes in it are adjacent. A set is dependent if at least two of the nodes in it is adjacent. A maximal independent set is an independent set which becomes dependent when any node is added to the set. The 4-cube has been examined.
Experimental results: solutions=2, BDDsize=30.

This paper is organized as follows: Section 2 describes the problem of generating all the r-partitions of n. Section 3 concludes the paper.

2. The problem of generating all the r-partitions of n

An r-partition of n is defined as a subdivision of all elements in the set $\{1,2,\ldots,n\}$ into r disjoint subsets $S_1, S_2, \ldots, S_r$. In this section, we consider the problem of generating all the r-partitions of n.

$$S_i \cap S_j = \emptyset \, (i \neq j), \quad S_i \neq \emptyset \, (1 \leqq i \leqq r), \quad \bigcup_{i=1}^{r} S_i = \{1,2,\ldots,n\}.$$

Therefore, for n=4, we have the following 15 partitions:

All the 1-partition of 4 is {1,2,3,4}.
All the 2-partitions of 4 are {1,2,3}{4},{1,2,4}{3},{1,2}{3,4},{1,3,4}{2}, {1,3}{2,4},{1,4}{2,3},{1}{2,3,4}.
All the 3-partitions of 4 are {1,2}{3}{4},{1,3}{2}{4},{1}{2,3}{4}, {1,4}{2}{3},{1}{2,4}{3},{1}{2}{3,4}.
All the 4-partition of 4 is {1}{2}{3}{4}.

It is well known that the number of all the r-partitions of n is the Stirling number of the second kind, $S(n,r)$ $(1 \leqq r \leqq n)$.

$$S(n,r) = \left(\sum_{i=0}^{r} (-1)^i \, {}_rC_i \, (r-i)^n \right) / r!$$

By the way, the (r+1)-partitions of n can be constructed from the r-partitions of n.

We assume that an r-partition of n consists of r subsets $S_1, S_2, \ldots, S_r$. By inserting the element n+1 into one of the r subsets, we can construct r r-partitions of n+1.

$$S_1 \cup \{n+1\}, S_2, S_3 \ldots, S_r$$
$$S_1, S_2 \cup \{n+1\}, S_3, \ldots, S_r$$
$$\cdot \cdot \cdot$$
$$S_1, S_2, S_3 \ldots, S_r \cup \{n+1\}$$

By adding a singleton {n+1} to an r-partition of n, an (r+1)-partition of n+1 is obtained.

$$S_1, S_2, S_3 \ldots, S_r, \{n+1\}$$

Thus, all the r-partitions of n can be represented by a tree.
For example, Figure 2.1 shows a tree corresponding to all the r-partitions of n ($1 \leqq n \leqq 4$, $1 \leqq r \leqq n$).

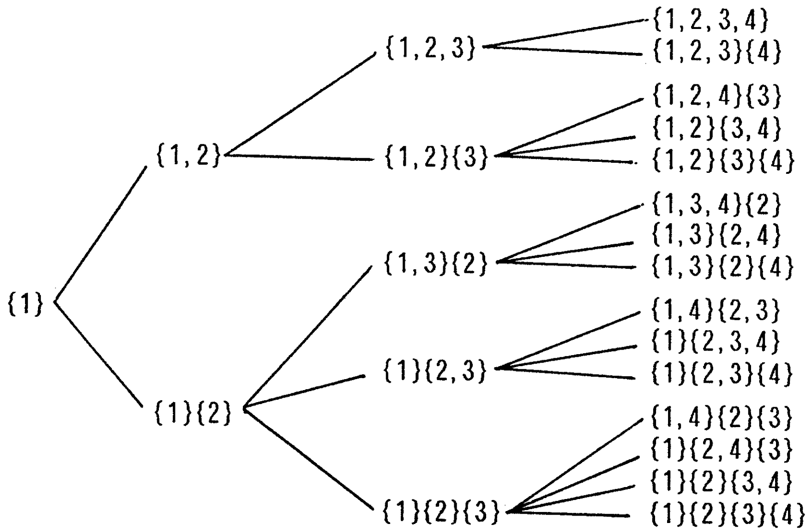
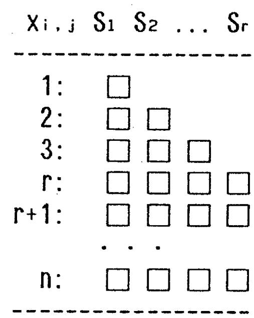


Figure 2.1. A tree corresponding to all the r-partitions of n ($1 \leqq n \leqq 4, 1 \leqq r \leqq n$).

Now, we will derive the Boolean function representing all the r-partitions of n. We define Boolean variables $x_{i,j}$ ($1 \leqq i \leqq n$, $1 \leqq j \leqq \min(i,r)$) as follows.

$x_{i,j}$ = 1, if the number i is included in the subset S_j.
0, if the number i is not included in the subset S_j.



We note that there are nr-r(r-1)/2 variables.

For example, for n=4 and r=3, the solution satisfying the equation $x_{1,1}\bar{x}_{2,1}, x_{2,2}x_{3,1}\bar{x}_{3,2}\bar{x}_{3,3}\bar{x}_{4,1}\bar{x}_{4,2}x_{4,3}=1$ is $x_{1,1}=1, x_{2,1}=0, x_{2,2}=1, x_{3,1}=1, x_{3,2}=0, x_{3,3}=0, x_{4,1}=0, x_{4,2}=0, x_{4,3}=1$. This means that the number 1 and 3 are included in S_1, the number 2 is included in S_2 and the number 4 is included in S_3. Thus, the above equation represents the 3-partition of 4, {1,3}{2}{4}.

Generally, the Boolean function representing all the r-partition of n is obtained as follows.

[Condition 1]

In an r-partition of n, the number i is contained in the subset S_j, that is , one of the subsets $S_1, S_2, \ldots, S_r$. This condition is described by Boolean variables as follows.

For any row i ($1 \leqq i \leqq n$), one variable in a row i is assigned 1 and other variables in a row i are assigned 0 among variables $x_{i,j}$ ($1 \leqq j \leqq \min(i,r)$). Thus, the Boolean formulas for this condition, A(i) ($1 \leqq i \leqq n$, $m=\min(i,r)$), are

$$A(1) = x_{1,1} = 1$$

$$A(i) = x_{i,1}\bar{x}_{i,2}\bar{x}_{i,3}\ldots\bar{x}_{i,m-1}\bar{x}_{i,m} + \bar{x}_{i,1}x_{i,2}\bar{x}_{i,3}\ldots\bar{x}_{i,m-1}\bar{x}_{i,m} + \ldots +$$
$$\bar{x}_{i,1}\bar{x}_{i,2}\bar{x}_{i,3}\ldots x_{i,m-1}\bar{x}_{i,m} + \bar{x}_{i,1}\bar{x}_{i,2}\bar{x}_{i,3}\ldots\bar{x}_{i,m-1}x_{i,m} = 1$$

[Condition 2]

In an r-partition of n, if the number i is included in the subset S_j, then at least one of the numbers 1,2,...,i-1 has to be included in the subset S_{j-1}. This condition is described by Boolean variables as follows.

If the variable $x_{i,j}$ is assigned 1, then at least one of the variables in a column j-1, $x_{k,j-1}$ ($j-1 \leqq k \leqq i-1$) is assigned 1.
Thus, the Boolean formulas for this condition, B(i,j) ($2 \leqq j \leqq r, j \leqq i \leqq n$), are

$$B(i,j) = \bar{x}_{i,j} + x_{j-1,j-1} + x_{j,j-1} + \ldots + x_{i-1,j-1} = 1$$

[Condition 3]

In an r-partition of n, at least one of the numbers r,r+1,...,n has to be contained in the subset S_r. This condition is described by Boolean variables as follows.
At least one of the variables $x_{i,r}$ ($r \leqq i \leqq n$) has to be assigned 1.
Thus, the Boolean formula for this condition, C(r), is

$$C(r) = x_{r,r} + x_{r+1,r} + \ldots x_{n,r} = 1$$

From condition 1, condition 2 and condition 3, we can obtain the Boolean function representing all the r-partitions of n.

$$\left(\prod_{i=1}^{n} A(i)\right) \cdot \left(\prod_{j=2}^{r} \prod_{i=j}^{n} B(i,j)\right) \cdot C(r) = 1$$

Experimental Results

For some n and r, we have computed S(n,r) and the number of nodes used to construct the BDD representing all the r-partitions of n and measured the running time required to construct the same BDD. The results are shown in Table 2.1.

n	r	S(n,r)	the number of nodes	running time
20	10	0.59175×10^{13}	1365	3.8
30	15	0.12879×10^{23}	4585	19.5
40	20	0.16218×10^{33}	10830	159.4
50	25	0.74538×10^{43}	21100	482.4

Table 2.1. S(n,r) and the number of nodes used to construct the BDD representing all the r-partitions of n and the running time required to construct the same BDD (times in seconds)

3. Conclusion

We have considered various combinatorial problems represented by Boolean functions. The problem of generating all the r-partitions of n has been examined in detail. It is surprising that a large number of combinatorial objects are represented by the BDD having a small number of nodes.

Acknowledgement

The authors would like to thank Mr.Shin-ichi Minato and Mr.Hiroyuki Ochi who offered them the Boolean function manipulatior. They would also like to express their sincere appreciation to Professor Naofumi Takagi, Mr.Kiyoharu Hamaguti and Mr.Yasuhiko Takenaga for valuable discussions and comments.

References

[1] S.Minato, N.Ishiura and S.Yajima: "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation", Proc. 27th ACM/IEEE DAC, pp.52-57, (June 1990).

[2] K.S.Brace, R.L.Rudell and R.E.Bryant: "Efficient Implementation of a BDD Package", Proc. 27th ACM/IEEE DAC, pp.40-45, (June 1990).

[3] H.Ochi, K.Yasuoka and S.Yajima: "A Breadth-First Algorithm for Efficient Manipulation of Shared Binary Decision Diagrams in the Secondary Memory", The 45th General Convention of IPSJ, 6-137, (Oct. 1992).

[4] C.L.Liu: "Introduction to Combinatorial Mathematics",McGraw-Hill,(1968).