Vol. 35 No. 9 Transactions of Information Processing Society of Japan Sep. 1994

Regular Paper

Combinatorial Algorithms Using Boolean Processing

ICHIRO SEMBA† and SHUZO YAJIMA††

The backtracking technique has been used to solve various problems of generating all combinatorial objects. A feature of this technique is that the conditions attached to the problem and the search for solutions are closely related. Thus, in order to obtain all solutions efficiently, it is necessary to discover suitable data structures for each problem. In this paper, we propose a new general technique for solving combinatorial problems by describing the conditions and searching for solutions separately. First, we describe the conditions attached to the problem by using Boolean functions. Next, we construct a binary decision diagram (BDD), representing Boolean functions by using an efficient BDD manipulator. Finally, we traverse the BDD and obtain all the solutions. By applying this technique to many combinatorial problems, we have discovered that the conditions attached to a problem can be briefly described by Boolean functions, and that all the solutions can be obtained efficiently.

1. Introduction

How to manipulate Boolean functions efficiently is an important problem in such applications as formal design verification, test generation, and logic synthesis. Since efficient manipulation and representation of Boolean functions are closely related, various methods of representing Boolean functions have been proposed.

A binary decision diagram (BDD) is a graph representation of Boolean functions.[1),2)] A shared binary decision diagram (SBDD) is an improved version of a BDD.[9),10)] Both have excellent properties for realizing efficient manipulation of Boolean functions. Recently, BDD manipulators have been implemented on workstations[12),17)] and widely used.

The backtracking technique has been used to solve various problems of generating combinatorial objects.[11)] A feature of this method is that the conditions attached to the problem and the search for solutions are closely related. Thus, in order to obtain satisfactory solutions, it is necessary to make efforts to discover suitable data structures for each combinatorial problem.

In this paper, we introduce a new technique for solving combinatorial problems by using BDD. This technique is essentially different from the backtracking technique. The application of BDD to combinatorial problems is discussed in this paper for the first time.

The technique separates the description of the conditions and the search for solutions. First, we describe the conditions attached to the problem by using Boolean functions. Next, we construct a binary decision diagram (BDD) representing Boolean functions by using an efficient BDD manipulator. Finally, we traverse the BDD and obtain all the solutions.

By applying this technique to various problems of generating combinatorial objects, we have discovered that the conditions attached a problem can be briefly described by Boolean functions, and that all the solutions can be obtained efficiently.

Since the description of the conditions and the search for solutions are separated in this technique, we can easily obtain solutions by changing the conditions. In other words, this technique is suitable for examining the solutions corresponding to various conditions within a short period of time.

This process corresponds to solving NP problems directly by using Boolean expressions. A BDD can be considered to be a new data structure or database.

Recently various applications of BDDs have attracted attention, and some interesting reports have been published.[3)–5),7),8),18)–22)]

One paper[22)] discusses combinatorial optimization problems that are represented by arithme-

† College of General Education, Ibaraki University
†† Department of Information Science, Faculty of Engineering, Kyoto University

tic expressions, and solves them by using the arithmetic Boolean expression manipulator BEM-II.[7]

This paper is organized as follows:

Section 2 describes the BDD and SBDD. Section 3 describes a fundamental generating algorithm (the algorithm generating all the r-combinations of the set $\{1, 2, \cdots, n\}$).[11),13),14)] Section 4 describes the following combinatorial problems:

1. The problem of generating all the r-permutations of the set $\{1, 2, \cdots, n\}$[11)]
2. The problem of generating all the r-partitions of the set $\{1, 2, \cdots, n\}$[11),15)]
3. The problem of generating all the balanced incomplete block designs[6)]

Section 5 describes the following graph problems:

1. The problem of generating all the graphs in which each node has some edges
2. The problem of finding all the paths of length r.

Section 6 concludes the paper.

2. Binary Decision Diagram (BDD) and Shared Binary Decision Diagram (SBDD)

We assume that a Boolean function with n variables $x_1, x_2, \cdots, x_n$ is denoted by $f(x_1, x_2, \cdots, x_n)$. When some variables x_i of function f with n variables $x_1, x_2, \cdots, x_n$ are replaced by 0(1), the function f is called a restriction of f.

Using the Shannon expansion,[16)] we can give a function f around variable x_i by

$$f(x_1, \cdots, x_i, \cdots, x_n) = x_i f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n) + \bar{x}_i f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n)$$

We now consider a graphical representation of a Boolean function $f(x_1, x_2, \cdots, x_n)$.

A Binary Decision Tree (BDT) is a binary tree with nodes of two types: terminal nodes and nonterminal nodes. Terminal nodes are labeled 0 or 1. Nonterminal nodes are labeled with one of the Boolean variables $x_1, x_2, \cdots, x_n$. Every nonterminal node has exactly two outgoing edges, which are labeled 0 and 1, and called the 0-edge and the 1-edge, respectively.

By repeating the Shannon expansion of a Boolean function $f(x_1, x_2, \cdots, x_n)$ recursively, we can derive a BDT corresponding to $f(x_1, x_2, \cdots, x_n)$. We note that a restriction of f is performed in decreasing order of the index of variables.

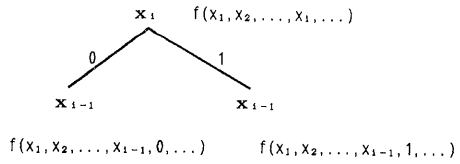


Fig. 1 Shannon expansion.

The process is as follows:

Let node x be labeled x_i, and let it correspond to a Boolean function $f(x_1, x_2, \cdots, x_i, \cdots)$. We assume that variables $x_{i+1}, \cdots, x_n$ have already been determined to be 0 or 1. A child of node x is labeled x_{i-1} and corresponds to $f(x_1, x_2, \cdots, x_{i-1}, 0, \cdots)$. An edge to the child is a 0-edge. Another child of node x is also labeled x_{i-1}, and corresponds to a Boolean function $f(x_1, x_2, \cdots, x_{i-1}, 1, \cdots)$. An edge to the child is a 1-edge. A graphical representation of this process in shown in **Fig. 1.** When the variables $x_1, x_2, \cdots, x_n$ have already been determined and the value of $f(x_1, x_2, \cdots, x_n)$ is 0(1), a child of node x is a terminal node labeled 0(1). We note that the root is labeled x_n and corresponds to a Boolean function $f(x_1, x_2, \cdots, x_n)$.

We note that the path from the root node to the terminal node labeled 1 corresponds to the solution satisfying the equation $f(x_1, x_2, \cdots, x_n) = 1$. If a 1(0)-edge emerges from a node labeled x_i in a path, the 1(0)-edge corresponds to $x_i = 1$ $(x_i = 0)$.

A Binary Decision Diagram (BDD) is defined as a directed acyclic graph obtained from a BDT by repeating the following transformations (1), (2), (3), and (4) until they are not applicable.

Transformation (1) : When both the 0-edge and the 1-edge point to the same terminal node, delete the nonterminal node and both the 0-edge and the 1-edge. The edge to this nonterminal node is changed so that it points to the terminal node.

Tranformation (2) : Share isomorphic subgraphs.

Transformation (3) : When both the 0-edge and the 1-edge of the nonterminal node x point to the same nonterminal node y, delete the nonterminal node x and both the 0-edge and the 1-edge. The edge to the nonterminal node x is changed so that it points to the nonter-

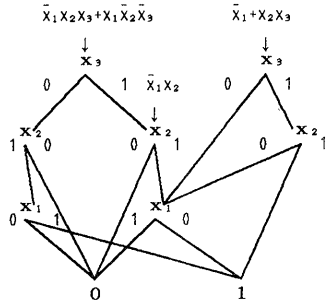


Fig. 2 SBDD representing three BDDs ($\bar{x}_1+x_2x_3$, $\bar{x}_1x_2$, $\bar{x}_1x_2x_3+x_1\bar{x}_2\bar{x}_3$).

minal node y.

Transformation (4) : Leave only one terminal node labeled 0(1).

We also note that different BDTs representing a Boolean function $f(x_1, x_2, \cdots, x_n)$ are constructed by following different orders of the index of variables. Therefore, different BDDs are also obtained.

A Shared Binary Decision Diagram (SBDD) is an improved version of a BDD. While a BDD represents a single Boolean function, an SBDD represents multiple Boolean functions by sharing sub-graphs of BDDs representing the same function.

An example in which an SBDD is constructed from three BDDs is shown in **Fig. 2**.

3. Fundamental Generating Algorithm (an algorithm for generating all the r-combinations of the set $\{1, 2, \cdots, n\}$)

In this chapter, we consider how to generate all the r-combinations of the set $\{1, 2, \cdots, n\}$ (r-combination of n for short).

[Definition]

An r-combination of n is defined as an unordered selection of r members of the set $\{1, 2, \cdots, n\}$.

The number of all the r-combinations of n is well known as the binomial coefficient, ${}_nC_r$ $(n!/(r!(n-r)!))$. We note that all the r-combinations of n are often used to generate other combinatorial objects.

We will now derive the Boolean function representing all the r-combinations of n. First, we define Boolean variables $x_1, x_2, \cdots, x_n$ as follows:

i:	1	2	3	4	5
x_i:	0	1	1	0	1

Fig. 3 Boolean variables representing a 3-combination of 5, 235.

[Boolean variables]

$x_i=1$, if the number i is contained in an r-combination of n.

0, if the number i is not contained in an r-combination of n.

As an example, a 3-combination of 5, 235, is represented in **Fig. 3**

Since the solution satisfying the equation $x_1x_2x_3\bar{x}_4\bar{x}_5=1$ is $x_1=1$, $x_2=1$, $x_3=1$, $x_4=0$, $x_5=0$, the equation $x_1x_2x_3\bar{x}_4\bar{x}_5=1$ corresponds to the 3-combination of 5, 123. Therefore, we can see that the following equation represents all the 3-combinations of 5:

$$\underset{123}{x_1x_2x_3\bar{x}_4\bar{x}_5}+\underset{124}{x_1x_2\bar{x}_3x_4\bar{x}_5}+\underset{125}{x_1x_2\bar{x}_3\bar{x}_4x_5}+\underset{134}{x_1\bar{x}_2x_3x_4\bar{x}_5}+\underset{135}{x_1\bar{x}_2x_3\bar{x}_4x_5}+\underset{145}{x_1\bar{x}_2\bar{x}_3x_4x_5}+\underset{234}{\bar{x}_1x_2x_3x_4\bar{x}_5}+\underset{235}{\bar{x}_1x_2x_3\bar{x}_4x_5}+\underset{245}{\bar{x}_1x_2\bar{x}_3x_4x_5}+\underset{345}{\bar{x}_1\bar{x}_2x_3x_4x_5}=1$$

We denote a Boolean function representing all the j-combinations of $i+j$ by $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$. The Boolean function $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$ satisfies the following recurrence relation:

[Boolean function]

Theorem 3. 1

$$f_{0,j}(x_1, x_2, \cdots, x_j)=x_1x_2\cdots x_j \quad (j\geqq 1) \tag{3. 1. 1}$$

$$f_{i,0}(x_1, x_2, \cdots, x_i)=\bar{x}_1\bar{x}_2\cdots\bar{x}_i \quad (i\geqq 1) \tag{3. 1. 2}$$

$$f_{i,j}(x_1, x_2, \cdots, x_{i+j}) = f_{i,j-1}(x_1, x_2, \cdots, x_{i+j-1})\,x_{i+j} + f_{i-1,j}(x_1, x_2, \cdots, x_{i+j-1})\,\bar{x}_{i+j} \quad (i\geqq 1,\ j\geqq 1) \tag{3. 1. 3}$$

Proof. Equations (3. 1. 1) and (3. 1. 2) are obvious. We will let U denote the set including all the j-combinations of $i+j$. The set U can be divided into two subsets V and W such that $V\cap W=\phi$, $V\cup W=U$. A combination included in the subset V must contain the value $i+j$. Thus, the Boolean function representing all the combinations in V is $f_{i,j-1}(x_1, x_2, \cdots, x_{i+j-1})\times x_{i+j}$. A combination included in the subset W does not contain the value $i+j$. Thus, the

Boolean function representing all the combinations in V is $f_{i-1,j}(x_1, x_2, \cdots, x_{i+j-1})\,\bar{x}_{i+j}$. From these facts, we can derive the recurrence relation (3. 1. 3).

Example: Boolean function $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$ $(1 \leqq i+j \leqq 3,\ 0 \leqq i,\ 0 \leqq j)$.

$f_{0,1}(x_1) = x_1$, $f_{1,0}(x_1) = \bar{x}_1$

$f_{0,2}(x_1, x_2) = x_1x_2$, $f_{1,1}(x_1, x_2) = \bar{x}_1x_2 + x_1\bar{x}_2$,

$f_{2,0}(x_1, x_2) = \bar{x}_1\bar{x}_2$

$f_{0,3}(x_1, x_2, x_3) = x_1x_2x_3$

$f_{1,2}(x_1, x_2, x_3) = x_1x_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2x_3$

$f_{2,1}(x_1, x_2, x_3) = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3$

$f_{3,0}(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3$

Thus, all the j-combinations of $i+j$ can be obtained by means of the following algorithm 3. 1:

Algorithm 3. 1

Step 1 : Construct the BDD representing all the j-combinations of $i+j$ by using the above recurrence relation.

Step 2 : Traverse all the paths of the BDD from the root corresponding to $f_{i,j}$ $(x_1, x_2, \cdots, x_{i+j})$ to the terminal node labeled 1.

We note that there is a one-to-one correspondence between the path from the root to the terminal node labeled 1 and the j-combination of $i+j$. If a 1-edge emerges from a node labeled x_i in a path, then the number i is contained in a j-combination of $i+j$. If a 0-edge emerges from a node labeled x_i in a path, then the number i is not contained in a j-combination of $i+j$.

Theorem 3. 2

The number of nodes used to construct the BDD representing the Boolean function $f_{i,j}$ $(x_1, x_2, \cdots, x_{i+j})$ is less than or equal to $ij+i+j$.

Proof. In order to obtain the Boolean function $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$, it is sufficient to compute the Boolean function $f_{u,v}(x_1, x_2, \cdots, x_{u+v})$ $(0 \leqq u \leqq i,\ 0 \leqq v \leqq j)$ by using the recurrence relation of Theorem 3. 1. Since the Boolean function $f_{u,v}(x_1, x_2, \cdots, x_{u+v})$ requires one node, we use $(i+1)(j+1)-1 = ij+i+j$ nodes to construct the BDD.

As an example, we show an SBDD representing Boolean functions $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$ $(1 \leqq i+j \leqq 3,\ 0 \leqq i,\ 0 \leqq j)$ in **Fig. 4**.

Lastly, we will mention two important combinatorial objects that are often used to generate other combinatorial objects.

One is the set of ways of selecting at least r members from the set $\{1, 2, \cdots, n\}$. The Boolean function representing these ways is

$$f_{n-r,r}(x_1, \cdots, x_n) + f_{n-r-1,r+1}(x_1, \cdots, x_n) + \cdots + f_{0,n}(x_1, \cdots, x_n) = 1$$

As an example, for $n=4$ and $r=2$,

$$\begin{aligned} &f_{2,2}(x_1, \cdots, x_4) + f_{1,3}(x_1, \cdots, x_4) + f_{0,4}(x_1, \cdots, x_4) \\ &= x_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3x_4 \\ &\quad + \bar{x}_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3x_4 \\ &\quad + x_1x_2x_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2x_3x_4 \\ &\quad + \bar{x}_1x_2x_3x_4 + x_1x_2x_3x_4 \end{aligned}$$

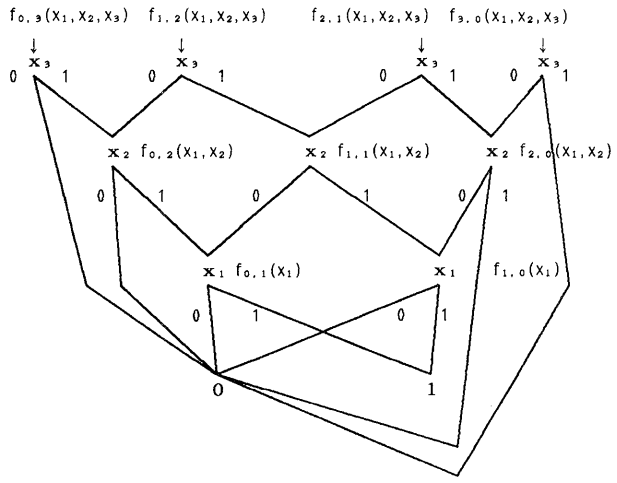


Fig. 4 SBDD representing Boolean functions $f_{i,j}(x_1, x_2, \cdots, x_{i+j})$ $(1 \leqq i+j \leqq 3,\ 0 \leqq i,\ 0 \leqq j)$.

$$= x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$
$$= 1$$

The other is the set of ways of selecting at most r members from the set $\{1, 2, \cdots, n\}$. The Boolean function representing these ways is

$$f_{n,0}(x_1, \cdots, x_n) + f_{n-1,1}(x_1, \cdots, x_n) + \cdots + f_{n-r,r}(x_1, \cdots, x_n) = 1$$

As an example, for $n=4$ and $r=1$

$$f_{3,1}(x_1, \cdots, x_4) + f_{4,0}(x_1, \cdots, x_4) = x_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_3\bar{x}_4 + \bar{x}_2\bar{x}_3\bar{x}_4 = 1$$

[Experimental results]

For some n are r, we have computed the number of nodes used to construct the BDD representing all the r-combinations of n. We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 1**.

It is interesting that a large number of combinations can be represented by a small number of nodes contained in the corresponding BDD. All the r-combinations of n are often used to generate other combinatorial objects.

Obviously, the number of nodes in BDD depends on the problem. Moreover, depending on the assignment of Boolean variables and the order of index of Boolean variables, the number of nodes in BDD varies greatly. Thus, the number of nodes in BDD reflects the complexity of the problem and the quality of the method.

Table 1 Number of Boolean variables and ${}_nC_r$ and the number of nodes used to construct the BDD representing all the r-combinations of n, and the running times required to construct the same BDD (times in seconds).

n	r	Number of variables	${}_nC_r$	Number of nodes	Running time
100	50	100	0.10089×10^{30}	2599	0.4
200	100	200	0.90549×10^{59}	10199	1.0
300	150	300	0.93759×10^{89}	22799	2.1
400	200	400	0.10295×10^{120}	40399	4.8
500	250	500	0.11674×10^{150}	62999	5.2
600	300	600	0.13511×10^{180}	90599	10.9
700	350	700	0.15857×10^{210}	123199	16.3
800	400	800	0.18804×10^{240}	160799	21.9

4. Various Combinatorial Problems

4.1 The Problem of Generating All the r-Permutations of the Set $\{1, 2, \cdots, n\}$

[Definition]

An r-permutation of the set $\{1, 2, \cdots, n\}$ (r-permutation of n, for short) is defined as an ordered arrangement of r members of the set $\{1, 2, \cdots, n\}$. The number of ways of arranging r out of n distinct objects is denoted by $P(n, r)$ and shown to be $n!/(n-r)!$.

[Boolean variables]

For $1 \leqq i \leqq r$, $1 \leqq j \leqq n$,

$x_{i,j} = 1$, if the i-th number is j.
0, if the i-th number is not j.

As an example, a 4-permutation of 5, 3152, is represented in **Fig. 5**.

[Condition]

[Condition 1]

In an r-permutation of n, the i-th number is one of the numbers $1, 2, \cdots, n$. This condition is described by Boolean variables as follows:

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

$$A(i) = f_{n-1,1}(x_{i,1}, x_{i,2}, \cdots, x_{i,n}) = 1$$

[Condition 2]

In an r-permutation of n, the number j appears at most once. The Boolean formulas for this condition, $B(j)$ ($1 \leqq j \leqq n$), are

$$B(j) = f_{r-1,1}(x_{1,j}, x_{2,j}, \cdots, x_{r,j}) + f_{r,0}(x_{1,j}, x_{2,j}, \cdots, x_{r,j}) = 1$$

[Boolean function]

From conditions 1 and 2, we can obtain the following Boolean function representing all the r-permutations of n:

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

$x_{i,j}$	1	2	3	4	5
1:	0	0	1	0	0
2:	1	0	0	0	0
3:	0	0	0	0	1
4:	0	1	0	0	0

Fig. 5 Boolean variables representing a 4-permutation of 5, 3152.

Table 2 The number of Boolean variables and $P(n, r)$ and the number of nodes used to construct the BDD representing all the r-permutations of n and the running time required to construct the same BDD (times in seconds).

n	r	Number of variables	$P(n, r)$	Number of nodes	Running time
1	1	1	1	1	
2	2	4	2	6	
3	3	9	6	23	
4	4	16	24	72	
5	5	25	120	201	
6	6	36	720	522	
7	7	49	5040	1291	
8	8	64	40320	3084	0.5
9	9	81	362880	7181	1.1
10	10	100	3628800	16398	2.8
11	11	121	39916800	36879	7.0

[Experimental results]

For some n and r, we have computed $P(n, r)$ and the number of nodes used to construct the BDD representing all the r-permutations of n. We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 2**.

4.2 The Problem of Generating All the r-Partitions of the Set $\{1, 2, \cdots, n\}$

[Definition]

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

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

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

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

[Basic idea]

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

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

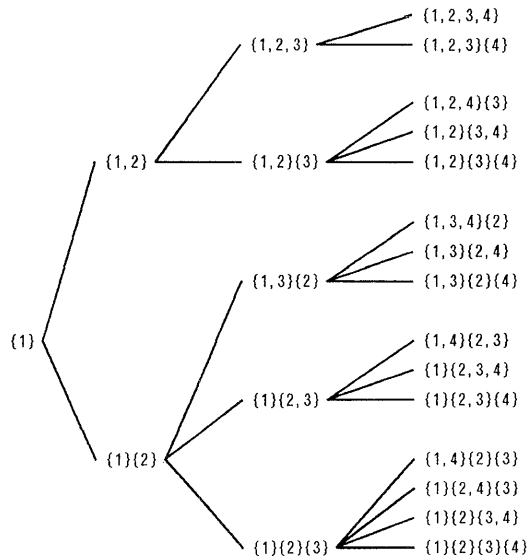


Fig. 6 Tree corresponding to all the r-partitions of n $(1 \leqq n \leqq 4, \ 1 \leqq r \leqq n)$.

$x_{i,j}$	S_1	S_2	S_3
1:	1		
2:	0	1	
3:	0	0	1
4:	1	0	0

Fig. 7 Boolean variables representing a 3-partition of 4, $\{1, 4\}\{2\}\{3\}$.

By adding a singleton $\{n+1\}$ to a r-partition of n, we obtain an $(r+1)$-partition of $n+1$:

$S_1, S_2, \cdots, S_r, \{n+1\}$

Thus, all the r-partitions of n can be represented by a tree, as shown by the example in **Fig. 6** $(1 \leqq n \leqq 4, \ 1 \leqq r \leqq n)$.

[Boolean variables]

For $1 \leqq i \leqq n$, $1 \leqq j \leqq \min(i, r)$,

$x_{i,j} = 1$, if the element i is included in the subset S_j.

0, if the element i is not included in the subset S_j.

As an example, a 3-partition of 4, $\{1, 4\}\{2\}\{3\}$, is shown in **Fig. 7**.

[Condition]

[Condition 1]

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

For any row i $(1 \leqq i \leqq n)$, only one variable in

a row i is assigned 1, and the others 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

$$A(i) = f_{m-1,1}(x_{i,1}, x_{i,2}, \cdots, x_{i,m}) = 1$$

[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 $j-1, j, \cdots, i-1$ 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

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

[Condition 3]

In an r-partition of n, at least one of the numbers $r, r+1, \cdots, n$ 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

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

[Boolean function]

From conditions 1, 2, and 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. We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 3**.

Table 3 The number of Boolean variables and $S(n, r)$ 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).

n	r	Number of variables	$S(n, r)$	Number of nodes	Running time
10	5	45	0.42525×10^{5}	170	0.2
20	10	155	0.59175×10^{13}	1365	2.6
30	15	345	0.12879×10^{23}	4585	22.6
40	20	610	0.16218×10^{33}	10830	103.2
50	25	950	0.74538×10^{43}	21100	350.7

4.3 The Problem of Generating All the Balanced Incomplete Block Designs

[Definition]

Let $A=\{a_1, a_2, \cdots, a_v\}$ be a set of v objects. A k-subset of A (a subset containing k objects of the set A) is called a block. A balanced incomplete block design of A (BIBD for short) is defined as a collection of b blocks, $B_1, B_2, \cdots, B_b$, satisfying the following three 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]

For $b=10$, $v=6$, $r=5$, $k=3$, $\lambda=2$, 12 solutions exist. One solution is as follows:

$B_1=\{a_1, a_2, a_3\}$
$B_2=\{a_1, a_2, a_4\}$
$B_3=\{a_1, a_3, a_5\}$
$B_4=\{a_1, a_4, a_6\}$
$B_5=\{a_1, a_5, a_6\}$
$B_6=\{a_2, a_3, a_6\}$
$B_7=\{a_2, a_4, a_5\}$
$B_8=\{a_2, a_5, a_6\}$
$B_9=\{a_3, a_4, a_5\}$
$B_{10}=\{a_3, a_4, a_6\}$

[Boolean variables]

For $1 \leqq i \leqq b$, $1 \leqq j \leqq v$,

$x_{i,j}$	a_1	a_2	a_3	a_4	a_5	a_6
B_1:	1	1	1	0	0	0
B_2:	1	1	0	1	0	0
B_3:	1	0	1	0	1	0
B_4:	1	0	0	1	0	1
B_5:	1	0	0	0	1	1
B_6:	0	1	1	0	0	1
B_7:	0	1	0	1	1	0
B_8:	0	1	0	0	1	1
B_9:	0	0	1	1	1	0
B_{10}:	0	0	1	1	0	1

Fig. 8 Boolean variables representing a solution $B_1=\{a_1, a_2, a_3\}$, $B_2=\{a_1, a_2, a_4\}$, $B_3=\{a_1, a_3, a_5\}$, $B_4=\{a_1, a_4, a_6\}$, $B_5=\{a_1, a_5, a_6\}$, $B_6=\{a_2, a_3, a_6\}$, $B_7=\{a_2, a_4, a_5\}$, $B_8=\{a_2, a_5, a_6\}$, $B_9=\{a_3, a_4, a_5\}$, $B_{10}=\{a_3, a_4, a_6\}$.

$x_{i,j}=1$, if an object a_j is contained in the block B_i.
$=0$, if an object a_j is not contained in the block B_i.

As an example, the above solution is represented in **Fig. 8**.

[Condition]

[Condition 1]

Each block $B_i(1\leqq i\leqq b)$ has k objects. We consider all the subsets with k objects. Thus, the Boolean formulas for this condition, $U(i)$ $(1\leqq i\leqq b)$, are

$$U(i)=f_{v-k,k}(x_{i,1}, x_{i,2}, \cdots, x_{i,v})=1$$

[Condition 2]

Each object $a_j(1\leqq j\leqq v)$ appears in exactly r of the b blocks. Thus, the Boolean formulas for this condition, $V(j)$ $(1\leqq j\leqq v)$, are

$$V(j)=f_{b-r,r}(x_{1,j}, x_{2,j}, \cdots, x_{b,j})=1$$

[Condition 3]

Every two objects appear simultaneously in exactly λ of the b blocks. Thus, the Boolean formulas for this condition, $W(j, m)$ $(1\leqq j<v, j<m\leqq v)$, are

$$W(j, m)=f_{b-\lambda,\lambda}(x_{1,j}x_{1,m}, x_{2,j}x_{2,m}, \cdots, x_{b,j}x_{b,m})=1$$

[Condition 4]

In order to avoid the repetition of the same balanced incomplete block design of A, we arrange $B_1, B_2, \cdots, B_b$ in lexicographical order. Since $B_i(1\leqq i\leqq b)$ are represented by $\{x_{i,1}, x_{i,2}, \cdots, x_{i,v}\}$, this condition is described by Boolean variables as follows:

For any $i(1\leqq i<b)$,

If $x_{i,1}\neq x_{i+1,1}$, then $x_{i,1}$ is assigned 1 and $x_{i+1,1}$ is assigned 0.

$$\overline{x_{i,1}\oplus x_{i+1,1}}+x_{i,1}\bar{x}_{i+1,1}=1$$

If $x_{i,1}=x_{i+1,1}$ and $x_{i,2}\neq x_{i+1,2}$, then $x_{i,2}$ is assigned 1 and $x_{i+1,2}$ is assigned 0.

$$x_{i,1}\oplus x_{i+1,1}+\overline{x_{i,2}\oplus x_{i+1,2}}+x_{i,2}\bar{x}_{i+1,2}=1$$

If $x_{i,1}=x_{i+1,1}$, $x_{i,2}=x_{i+1,2}, \cdots, x_{i,v-1}=x_{i+1,v-1}$ and $x_{i,v}\neq x_{i+1,v}$ then $x_{i,v}$ is assigned 1 and $x_{i+1,v}$ is assigned 0.

$$x_{i,1}\oplus x_{i+1,1}+x_{i,2}\oplus x_{i+1,2}+\cdots+x_{i,v-1}\oplus x_{i+1,v-1}+\overline{x_{i,v}\oplus x_{i+1,v}}+x_{i,v}\bar{x}_{i+1,v}=1$$

Thus, the Boolean formulas for this condition, $Z(i)$ $(1\leqq i<b)$ are

$$Z(i)=\prod_{m=1}^{v}\left(\sum_{j=1}^{m-1}(x_{i,j}\oplus x_{i+1,j})+\overline{x_{i,m}\oplus x_{i+1,m}}+x_{i,m}\bar{x}_{i+1,m}\right)=1$$

Table 4 Numbers of Boolean variables, of BIBDs, and of nodes used to construct the BDD representing all the BIBDs, and the running times required to construct the same BDD (times in seconds).

b	v	r	k	λ	Number of variables	Number of BIBDs	Number of nodes	Running time
10	6	5	3	2	60	12	463	0.7
12	9	4	3	1	108	840	23922	16.8
12	9	8	6	5	108	840	23737	113.0
7	7	3	3	1	49	30	622	0.7
7	7	4	4	1	49	30	627	0.5

[Boolean function]

From conditions 1, 2, 3, and 4, we can obtain the Boolean function representing all the balanced incomplete block designs.

$$\left(\prod_{i=1}^{b}U(i)\right)\cdot\left(\prod_{j=1}^{v}V(j)\right)\cdot\left(\prod_{j=1}^{v-1}\prod_{m=j+1}^{v}W(j, m)\right)\cdot\left(\prod_{i=1}^{b-1}Z(i)\right)=1$$

[Experimental results]

For some b, v, r, k, λ, we have computed the number of balanced incomplete block designs and the number of nodes used to construct the BDD representing all the balanced incomplete block designs.

We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 4**.

5. Various Graph Problems

5.1 The Problem of Generating All the Graphs in which Each Node Has Some Edges

[Definition]

The number of nodes is denoted by n. We consider how to generate all the undirected graphs with the restriction that node $i(1\leqq i\leqq n)$ has n_i edges. We assume that the edge $a=(i, j)$ $(i\neq j)$.

[Example]

For $n=5$, $n_1=1$, $n_2=2$, $n_3=2$, $n_4=2$, $n_5=1$: There are seven graphs.

The solutions are shown in **Fig. 9**.

[Boolean variables]

For $1\leqq i\leqq n$, $1\leqq j\leqq n$,

$x_{i,j}=1$, if node i and node j are adjacent.

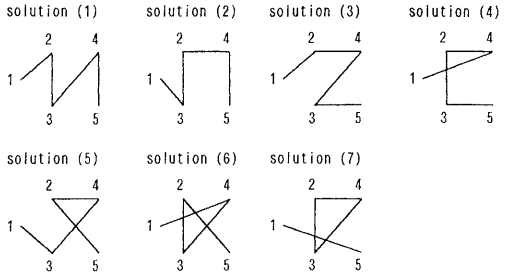


Fig. 9 The solutions of an example.

$x_{i,j}$	1	2	3	4	5
1:	0	1	0	0	0
2:	1	0	1	0	0
3:	0	1	0	1	0
4:	0	0	1	0	1
5:	0	0	0	1	0

Fig. 10 Boolean variables representing a solution (1) in Fig. 9.

0, if node i and node j are not adjacent.

As an example, the solution (1) in Fig. 9 is represented in **Fig. 10**.

[**Condition**]

[Condition 1]

No edges (i, i) $(1 \leqq i \leqq n)$ exist. The Boolean formulas for this condition, $A(i)$ $(1 \leqq i \leqq n)$, are

$$A(i) = \bar{x}_{i,i} = 1$$

[Condition 2]

From the property of the undirected graph, we have the condition that $x_{i,j} = x_{j,i}$. The Boolean formulas for this condition, $B(i, j)$ $(1 \leqq i < n,\ i < j \leqq n)$, are

$$B(i, j) = \overline{x_{i,j} \oplus x_{j,i}} = 1$$

[Condition 3]

Each node $i (1 \leqq i \leqq n)$ has n_i edges. This condition is equivalent to n_i-combinations of $\{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$. Thus, the Boolean formulas for this condition $C(i, n_i)$ $(1 \leqq i \leqq n)$, are

$$C(i, n_i) = f_{n-n_i,n_i}(x_{i,1}, x_{i,2}, \cdots, x_{i,n}) = 1$$

[**Boolean function**]

From conditions 1, 2, and 3, we can obtain the Boolean function representing all the graphs with the restriction that each node has some edges.

Table 5 Numbers of Boolean variables, of graphs satisfying the condition that $n_i = r$ $(1 \leqq i \leqq n)$, and of nodes used to construct the BDD, and the running times required to construct the same BDD (times in seconds).

n	r	Number of variables	Number of graphs	Number of nodes	Running time
8	1	28	105	291	
8	2	28	3507	1628	0.3
8	3	28	19355	3684	0.4
8	4	28	19355	3684	0.4
8	5	28	3507	1628	0.3
8	6	28	105	291	
8	7	28	1	28	
9	1	36	0	0	
9	2	36	30016	4596	0.7
9	3	36	0	0	1.3
9	4	36	1024380	22102	1.7
9	5	36	0	0	1.4
9	6	36	30016	4596	0.7
9	7	36	0	0	
9	8	36	1	36	
10	1	45	945	1080	0.3
10	2	45	286884	12197	2.0
10	3	45	11180820	59070	6.2
10	4	45	66462606	117055	9.5
10	5	45	66462606	117055	10.1
10	6	45	11180820	59070	6.6
10	7	45	286884	12197	2.3
10	8	45	945	1080	0.4
10	9	45	1	45	

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

[**Experimental results**]

For some n and k, we have computed the number of graphs satisfying the condition that $n_i = k$ $(1 \leqq i \leqq n)$, and the number of nodes used to construct the BDD. We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 5**.

5.2 The Problem of Finding All the Paths of Length r

[**Definition**]

It is assumed that an undirected graph $G = (N, E)$, where each edge has a weight 1, is given. This problem is to find all the paths of length r from node s (a starting node) to node d (a destination node). We note that a path does not contain any loops.

[Example]

For a graph $G=(N, E)$, $N=\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $E=\{(1, 2), (1, 4), (2, 3), (2, 5), (3, 6), (4, 5), (4, 7), (5, 6), (5, 8), (6, 9), (7, 8), (8, 9)\}$, $s=1$, $d=9$.

All the paths of length 6 from node $s(=1)$ to node $d(=9)$ are

(1, 2, 5, 4, 7, 8, 9), (1, 2, 3, 6, 5, 8, 9),
(1, 4, 7, 8, 5, 6, 9), (1, 4, 5, 2, 3, 6, 9).

They are shown in **Fig. 11.**

[Boolean variables]

For $0\leqq i\leqq r$, $1\leqq j\leqq n$,

$x_{i,j}=1$, if the i-th node in the path is node j.
0, if the i-th node in the path is not node j.

We note that the 0th node is a starting node.

As an example, a path of length 6 from 1 to 9, (1, 2, 5, 4, 7, 8, 9) is shown in **Fig. 12**,

[Condition]

[Condition 1]

The starting point is node s. Thus, the variable $x_{0,s}$ is assigned 1, and the other variables $x_{0,j}(1\leqq j\leqq s-1,\ s+1\leqq j\leqq n)$ are assigned 0. The Boolean formula for this condition, $A(n)$, is as follows:

$$A(n)=\bar{x}_{0,1}\bar{x}_{0,2}\cdots\bar{x}_{0,s-1}x_{0,s}\bar{x}_{0,s+1}\cdots\bar{x}_{0,n}=1$$

[Condition 2]

The destination is node d. Thus, the variable $x_{r,d}$ is assigned 1 and other variables $x_{r,j}(1\leqq j\leqq d-1,\ d+1\leqq j\leqq n)$. The Boolean formula for this condition, $B(n)$, is as follows:

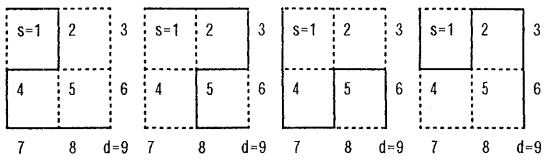
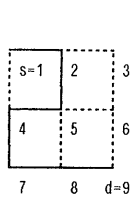


Fig. 11 All the paths of length 6 from 1 to 9.



$x_{i,j}$	1	2	3	4	5	6	7	8	9
0:	1	0	0	0	0	0	0	0	0
1:	0	1	0	0	0	0	0	0	0
2:	0	0	0	0	1	0	0	0	0
3:	0	0	0	1	0	0	0	0	0
4:	0	0	0	0	0	0	1	0	0
5:	0	0	0	0	0	0	0	1	0
6:	0	0	0	0	0	0	0	0	1

Fig. 12 Boolean variables representing a path of length 6 from 1 to 9, (1, 2, 5, 4, 7, 8, 9).

$$B(n)=\bar{x}_{r,1}\bar{x}_{r,2}\cdots\bar{x}_{r,d-1}x_{r,d}\bar{x}_{r,d+1}\cdots\bar{x}_{r,n}=1$$

[Condition 3]

The i-th $(1\leqq i\leqq r-1)$ node in the path of length r is selected from nodes $j(1\leqq j\leqq n)$. The Boolean formula for this condition, $C(i)$ $(1\leqq i\leqq r-1)$, is as follows:

$$C(i)=f_{n-1,1}(x_{i,1}, x_{i,2}, \cdots, x_{i,n})=1$$

[Condition 4]

The node j is included in a path of length r at most once.

The Boolean formula for this condition, $D(j)$ $(1\leqq j\leqq n)$, is as follows:

$$D(j)=f_{r,1}(x_{0,j}, x_{1,j}, \cdots, x_{r,j})+f_{r+1,0}(x_{0,j}, x_{1,j}, \cdots, x_{r,j})=1$$

[Condition 5]

We denote the set of nodes adjacent to node j by adj(j). Let adj$(j)=\{j_1, j_2, \cdots, j_m\}$. One node must be selected from the set adj(j). Therefore, if the i-th node is node j, then the $(i+1)$th node is selected from the set adj(j).

The Boolean formulas for this condition, $E(i, j)$ $(0\leqq i\leqq r-1,\ 1\leqq j\leqq n)$, are as follows:

$$E(i, j)=\bar{x}_{i,j}+f_{m-1,1}(x_{i+1,j_1}, x_{i+1,j_2}, \cdots, x_{i+1,j_m})=1$$

[Boolean function]

From conditions 1, 2, 3, 4, and 5, we can obtain the Boolean function representing all the paths of length r:

$$A(n)\cdot B(n)\cdot\left(\prod_{i=1}^{r-1}C(i)\right)\cdot\left(\prod_{j=1}^{n}D(j)\right)\cdot\left(\prod_{i=0}^{r-1}\prod_{j=1}^{n}E(i, j)\right)=1$$

[Experimental results]

We have considered how to find all the paths of length r from the starting point (1, 1) to the destination point (m, n) in **Fig. 13**. It is assumed that the distance between points is 1.

The shortest path is defined as a path of length r such that no path of length $t(t<r)$ exists.

For some m and n, we have examined the number of all the shortest paths and the number

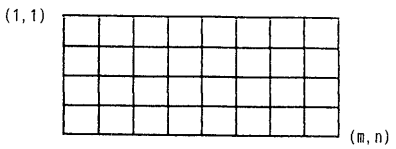


Fig. 13 Sample graph.

Table 6 The number of Boolean variables and the number of all the shortest paths and the number of nodes used to construct the BDD representing all the shortest paths and the running time required to construct the same BDD (times in seconds).

m	n	r	Number of variables	Number of shortest paths	Number of nodes	Running time
2	2	2	12	2	12	
3	3	4	45	6	80	
4	4	6	112	20	268	0.7
4	5	7	160	35	426	1.3
4	6	8	216	56	620	2.9
5	5	8	225	70	666	3.4
5	6	9	300	126	968	6.3
5	7	10	385	210	1326	13.0
6	6	10	396	252	1388	15.1
6	7	11	504	462	1900	26.8
6	8	12	624	792	2492	53.1
7	7	12	637	924	2572	57.5
7	8	13	728	1716	3372	91.9
7	9	14	882	3003	4280	164.6
8	8	14	960	3432	4380	151.2

Table 7 The number of Boolean variables and the number of all the longest paths and the number of nodes used to construct the BDD representing all the longest paths and the running time required to construct the same BDD (times in seconds).

m	n	r	Number of variables	Number of longest paths	Number of nodes	Running time
2	2	2	12	2	12	
3	3	8	81	2	140	0.4
3	4	11	144	4	386	1.4
2	7	13	196	1	196	2.7
3	5	14	225	8	859	4.6
4	4	14	224	32	1491	5.0
2	8	14	240	8	444	4.5
2	9	17	324	1	324	13.3
3	6	17	324	16	1670	15.7
2	9	17	324	1	324	13.4
2	10	18	380	10	716	19.4
4	5	19	400	20	3152	29.0
3	7	20	441	32	2955	38.0
2	11	21	484	1	484	39.1
2	12	22	506	12	1052	56.7
4	6	22	552	256	11482	77.8
3	8	23	576	64	4874	81.0

Table 8 Numbers of Boolean variables, of Hamilton cycles, and of nodes used to construct the BDD representing all the Hamilton cycles, and the running times required to construct the same BDD (times in seconds).

m	n	r	Number of variables	Number of Hamilton cycles	Number of nodes	Running time
2	2	4	16	2	16	
3	3	9	81	0	0	0.4
3	4	12	144	4	215	1.5
3	5	15	225	0	0	4.2
4	4	16	256	12	779	6.7
3	6	18	324	8	753	14.5
2	10	20	400	2	400	23.9
4	5	20	400	28	2095	25.5
3	7	21	441	0	0	27.1
2	11	22	484	2	484	39.6
2	12	24	576	2	576	68.7
3	8	24	576	16	2153	83.4
4	6	24	576	74	5460	80.7
5	5	25	625	0	0	64.6
2	13	26	676	2	676	107.9
3	9	27	729	0	0	108.4

of nodes used to construct the BDD representing all the shortest paths. We have also measured the running time required to construct the same BDD, coded in C, on a Sony bigNEWS workstation (16 MB). The results are shown in **Table 6**.

The longest path is defined as a path of length r such that no path of length $t(t>r)$ exists.

For some m and n, we have examined the number of all the longest paths. The results are shown in **Table 7**.

Since a cycle of length r is equal to a path of length r such that the starting node is the same as the destination node, we can find all the cycles of length r by using this Boolean function with some modifications. We have computed all the Hamilton cycles in Fig. 13. The results are shown in **Table 8**.

6. Conclusion

We have proposed a new technique for solving combinatorial problems by separating the description of the conditions and the search for solutions. This technique is essentially different from backtracking.

First, we describe the conditions attached to the problem by using Boolean functions. Next, we construct a binary decision diagram (BDD)

representing Boolean functions by using an efficient BDD manipulator. Finally we traverse the BDD and obtain all the solutions.

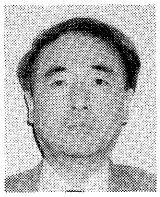
By applying this technique to many combinatorial problems, we have discovered that the conditions attached to the problem can be briefly described by Boolean functions, and that all the solutions can be obtained efficiently.

Acknowledgements The authors would like to thank Prof. Hiromi Hiraishi, Mr. Shin-ichi Minato, and Dr. Hiroyuki Ochi, who provided them with a Boolean function manipulator. They would also like to express their sincere appreciation to Prof. Naofumi Takagi, Dr. Kiyoharu Hamaguti, and Mr. Yasuhiko Takenaga for valuable discussions and comments.

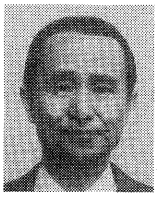
References

1) Akers, S. B.: Binary Decision Diagrams, *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 509-516 (1978).
2) Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 667-691 (1985).
3) Fujita, M. and Clarke, E. M.: Application of BDD to CAD for Digital Systems, *J. IPS Japan*, Vol. 34, No. 5, pp. 609-616 (1993) (in Japanese).
4) Ishiura, N.: An Introduction to Binary Decision Diagrams, *J. IPS Japan*, Vol. 34, No. 5, pp. 585-592 (1993) (in Japanese).
5) Kimura, S.: Parallel Binary Decision Diagram Manipulation, *J. IPS Japan*, Vol. 34, No. 5, pp. 623-630 (1993) (in Japanese).
6) Liu, C. L.: *Introduction to Combinatorial Mathematics*, McGraw-Hill (1968).
7) Minato, S.: BEM-II: An Arithmetic Boolean Expression Manipulator Using Binary Decision Diagrams, *Technical Report of the IEICE*, COMP92-75 (1993) (in Japanese).
8) Minato, S.: Technique for BDD Manipulation on Computers *J. IPS Japan*, Vol. 34, No. 5, pp. 593-599 (1993) (in Japanese).
9) Minato, S., Ishiura, N. and Yajima, S.: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, *Proc. 27th ACM/IEEE DAC*, pp. 52-57 (June 1990).
10) Minato, S., Ishiura, N. and Yajima, S.: Shared Binary Decision Diagrams for Efficient Boolean Function Manipulation, *Trans. of IPSJ*, Vol. 32, No. 1, pp. 77-85 (1991) (in Japanese).
11) Nijenhuis, A. and Wilf, H. S.: *Combinatorial Algorithms*, Academic Press (1978).
12) Ochi, H., Yasuoka, K. and Yajima, S.: A Breadth-First Algorithm for Efficient Manipulation of Shared Binary Decision Diagrams in the Secondary Memory, *45th General Convention of the IPSJ*, 6-137 (Oct. 1992).
13) Semba, I.: A Note on Enumerating Combinations in Lexicographical Order, *Journal of Information Processing*, Vol. 4, No. 1, pp. 35-37 (1981).
14) Semba, I.: An Efficient Algorithm for Generating All k-Subsets ($1 \leqq k \leqq m \leqq n$) of the Set $\{1, 2, \cdots, n\}$ in Lexicographical Order, *Journal of Algorithms*, 5, pp. 281-283 (1984).
15) Semba, I.: An Efficient Algorithm for Generating All Partitions of the Set $\{1, 2, \cdots, n\}$, *J. of Inf. Process.*, Vol. 7, No. 1, pp. 41-42 (1984).
16) Shannon, C. E.: A Symbolic Analysis of Relay and Switching Circuits, *Trans. AIEE*, Vol. 57, pp. 713-723 (1938).
17) Brace, K. S., Rudell, R. L. and Bryant, R. E.: Efficient Implementation of a BDD Package, *Proc. 27th ACM/IEEE DAC*, pp. 40-45 (June 1990).
18) Semba, I. and Yajima, S.: 組合せ問題の論理関数による解法について, 数理解析研究所講究録 833, 計算機構とアルゴリズム, pp. 204-213 (1993).
19) Semba, I. and Yajima, S.: Combinatorial Algorithms by Boolean Processing I, 情報処理学会研究報告, Vol. 93, No. 24, pp. 25-32 (1993).
20) Semba, I. and Yajima, S.: Combinatorial Algorithms by Boolean Processing II, 情報処理学会研究報告, Vol. 93, No. 24, pp. 33-40 (1993).
21) Watanabe, Y. and Kukimoto, Y.: Applications of Binary Decision Diagrams, *J. IPS Japan*, Vol. 34, No. 5, pp. 600-608 (1993) (in Japanese).
22) Yanagiya, M.: Combinatorial Optimization via BDD-Based Procedures, *J. IPS Japan*, Vol. 34, No. 5, pp. 617-623 (1993) (in Japanese).

(Received April 8, 1993)
(Accepted May 12, 1994)



Ichiro Semba was born on October 7, 1948. He graduated from the Department of Pure and Applied Science, College of General Education, University of Tokyo in 1972. He received Master course in coordinated science from University of Tokyo in 1974. Since 1993, he has been a Professor in College of General Education of Ibaraki University. His current research interests include combinatorial algorithms and Boolean processing. He is a member of the Information Processing Society of Japan and the Mathematical Society of Japan.



Shuzo Yajima was born in Takarazuka, Japan, on December 6, 1933. He received the B. E., M. E., and Ph. D. degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1956, 1958, and 1964, respectively. He developed Kyoto University's first digital computer, KDC-I, in 1960. In 1961 he joined the faculty of Kyoto University. Since 1971 he has been a Professor in the Department of Information Science, Faculty of Engineering, Kyoto University, engaged in research and education in logic circuit, switching, and automata theory. Dr. Yajima was a Trustee of the Institute of Electronics and Communication Engineers of Japan and Chairman of the Technical Committee on Automata and Languages of the Institute. He served on the Board of Directors of the Information Processing Society of Japan.