A Reduction Method for the Number of Variables to Represent Index Generation Functions: s-Min Method

Tsutomu Sasao
Dept. of Computer Science, Meiji University
Kawasaki, Kanagawa 214-8571, Japan

***Abstract*—Most n-variable incompletely specified index generation functions with weight k can be represented by fewer variables than n when $k \ll 2^n$. Furthermore, with a linear decomposition, the function can be represented by still fewer variables. In this paper, we propose an iterative improvement method, called the s-Min method, to reduce the number of variables.**

***Keywords*-incompletely specified function, index generation function, functional decomposition, linear transformation, iterative improvement.**

I. INTRODUCTION

Index generation functions [3], [4] are useful for computer virus scanners and the routing of packets across the internet. In these applications, functions must be updated frequently. Thus, index generation functions are often implemented by memory.

To reduce the total cost of realizing index generation functions, the **linear decomposition** shown in Fig. 1.1 is effective [6]. In Fig. 1.1, L realizes a linear function, while G realizes a general function. L is implemented by a circuit consisting of EXOR gates, while G is implemented by a memory.

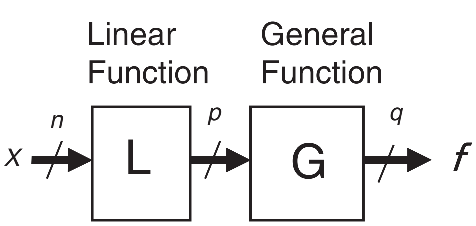

Figure 1.1. Linear Decomposition.

When a given function f is defined for only k input combinations and $k \ll 2^n$, in most cases, p, the number of variables for G in Fig. 1.1 can be smaller than n. We assume that the cost of L is proportional to np, while the cost of G is proportional to $q2^p$, where $q \leq p \leq n$, and $q = \lceil \log_2(k+1) \rceil$.

In this paper, we try to find a linear decomposition

$$f(x_1, x_2, \ldots, x_n) = g(y_1, y_2, \ldots, y_p),$$

that minimizes p, where y_j $(j = 1, 2, \ldots, p)$ are **linear functions** of x_i $(i = 1, 2, \ldots, n)$. Since the search space for the linear functions is very large, to obtain an optimal solution is hard. So, in this paper, we introduce a local search method called the **s-Min method**. The s-Min method iteratively replaces a set of linear functions with another set of linear functions to reduce the number of the variables in the decomposition. A similar method is presented in [5].

The rest of the paper is organized as follows: Section II introduces index generation functions; Section III introduces collision degree and shows its properties; Section IV shows local search algorithms called s-Min methods; Section V illustrate the algorithms using examples; Section VI shows experimental results; Section VII compares this method with other methods; and Section VIII summarizes the paper.

II. INDEX GENERATION FUNCTION

This section introduces an index generation function and its basic properties.

Definition 2.1: Consider a set of k different vectors of n bits. These vectors are called **registered vectors**. For each registered vector, assign a unique integer from 1 to k. A **registered vector table** shows an **index** for each registered vector. The value of an **incompletely specified index generation function** is a corresponding index when the input equals to a registered vector, and undefined (d, *don't care*) otherwise. The incompletely specified index generation function is a mapping $M \rightarrow \{1, 2, \ldots, k\}$, where $M \subset B^n$ is a set of registered vectors, and $B = \{0, 1\}$. k is the **weight** of the function.

Definition 2.2: A **compound variable** has a form $y = c_1x_1 \oplus c_2x_2 \oplus \cdots \oplus c_nx_n$, where $c_i \in \{0, 1\}$, and $\oplus$ denotes the mod 2 addition. The **compound degree** of the variable y is $\sum_{i=1}^{n} c_i$, where $\sum$ denotes integer addition, and the c_i's are viewed as integers. When the compound degree is 1, y is a single variable x_i, and is called **primitive**.

Note that compound variables are linear functions of $x_1, x_2, \ldots, x_n$.

From here, both primitive variables and compound variables are often called **variables**.

III. COLLISION DEGREE AND ITS PROPERTIES

In this section, to find a good linear decomposition, we introduce a *partial vector* and a *collision degree*.

0195-623X/15 $31.00 © 2015 IEEE
DOI 10.1109/ISMVL.2015.40

Table 3.1
INDEX GENERATION FUNCTION.

x_1	x_2	x_3	x_4	f
0	0	0	1	1
0	1	0	0	2
1	0	0	0	3
1	1	0	0	4

Definition 3.1: Let $f(X)$ be an incompletely specified index generation function, where $X = \{x_1, x_2, \ldots, x_n\}$ is the set of variables in f. Let X_1 be a proper subset of X. Let $\vec{X_1}$ be an ordered set of X_1. Then, $\vec{X_1}$ is a **partial vector** of X. Suppose that the values of $\vec{X_1}$ are fixed at $\vec{a} = (a_1, a_2, \ldots, a_s)$, where $a_i \in B$. Let $N(f, \vec{X_1}, \vec{a})$ be the number of the registered vectors such that the value of f is non-zero. Then, the **collision degree** is

$$CD(f : X_1) = \max_{\vec{a} \in B^s} \left\{ N(f : \vec{X_1}, \vec{a}) \right\},$$

where s denotes the number of variables in X_1.

Example 3.1: Consider the index generation function f shown in Table 3.1. We have:

$$\begin{aligned} N(f : (x_1, x_2), (0,0)) &= |\{1\}| = 1, \\ N(f : (x_2, x_4), (1,0)) &= |\{2,4\}| = 2, \\ N(f : (x_2, x_4), (0,1)) &= |\{1\}| = 1, \\ N(f : (x_4), (0)) &= |\{2,3,4\}| = 3. \end{aligned}$$

■

Lemma 3.1: Consider the decomposition chart of $f(X)$, where X_1 denotes the column variables and $X - X_1$ denote the row variables. Then, the collision degree $CD(f : X_1)$ denotes the maximal number of non-zero elements in the columns.

Example 3.2: Fig. 3.1 shows a decomposition chart of the index generation function shown in Table 3.1. In this chart, the column variables are $X_1 = \{x_2, x_4\}$, and blank elements show *don't cares*. The number of non-zero elements are, from the left to the right, 1,1,0,2. Note that the rightmost column has the maximum number of non-zero elements in a column, 2, when $(x_2, x_4) = (1, 0)$. Thus, $CD(f : \{x_2, x_4\}) = 2$. ■

Example 3.3: Consider the index generation function f shown in Table 3.1. We have:

$$\begin{aligned} CD(f : \{x_1, x_2\}) &= Max\{|\{1\}|, |\{2\}|, |\{3\}|, |\{4\}|\} = 1. \\ CD(f : \{x_2, x_4\}) &= Max\{|\{1\}|, |\{2,4\}|, |\{3\}|\} = 2. \\ CD(f : \{x_1\}) &= Max\{|\{1,2\}|, |\{3,4\}|\} = 2. \\ CD(f : \{x_2\}) &= Max\{|\{1,3\}|, |\{2,4\}|\} = 2. \\ CD(f : \{x_3\}) &= Max\{|\{1,2,3,4\}|, |\phi|\} = 4. \\ CD(f : \{x_4\}) &= Max\{|\{2,3,4\}|, |\{1\}|\} = 3. \end{aligned}$$

■

An incompletely specified index generation function $f(X)$ can be represented by a subset X_1 of X if every assignment of values of a registered vector to the variables X_1 uniquely specifies the value of f.

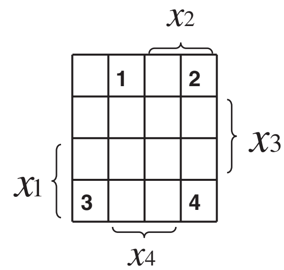

Figure 3.1. Decomposition Chart of an Index Generation Function.

Theorem 3.1: Let $f(X)$ be an incompletely specified index generation function. f can be represented as a function of X_1, where X_1 is a proper subset of X if

$$CD(f : X_1) = 1.$$

(Proof) Consider the decomposition chart, where X_1 denotes the column variables. If $CD(f : X_1) = 1$, then each column has at most one non-zero element. In this case, the function can be represented with only the column variables [4]. □

Example 3.4: Consider the index generation function shown in Table 3.1. Since If $CD(f : \{x_1, x_2\}) = 1$, the function can be represented with only x_1 and x_2. If fact, the function can be represented as

$$f = 1 \cdot \bar{x}_1 \bar{x}_2 \vee 2 \cdot \bar{x}_1 x_2 \vee 3 \cdot x_1 \bar{x}_2 \vee 4 \cdot x_1 x_2.$$

■

Theorem 3.2: Let $f(X)$ be an incompletely specified index generation function. Let X_1 be a proper subset of X. Then, to represent $f(X)$, at least $\lceil \log_2 CD(f : X_1) \rceil$ compound variables are necessary in addition to the variables in X_1.

(Proof) When $CD(f : X_1) = a$, a registered vectors are indistinguishable. To distinguish these vectors, we need at least $\lceil \log_2 a \rceil$ variables in addition to the variables in X_1. □

Corollary 3.1: Let $f(X)$ be an incompletely specified index generation function, and let X_1 be a proper subset of X. A necessary condition that f be represented by X_1 and one compound variable is

$$CD(f : X_1) = 2.$$

Corollary 3.2: Let $f(X)$ be an incompletely specified index generation function, and let X_1 be a subset of X. A necessary condition that f be represented by X_1 and a pair of compound variables is

$$CD(f : X_1) \leq 4.$$

IV. s-Min Method

A travelling salesman problem (TSP) is a combinatorial optimization problem whose search space is large. A method to obtain a locally optimal solution for a TSP, **2-Opt method** is known. In the 2-Opt method, a pair of edges of the current solution is replaced with an another pair of edges, and a new network is produced. An improved solution may be found in a new network, and a locally optimal solution can be obtained.

In a similar manner, in the **s-Min method**, an arbitrary set of s variables in X_1 is replaced with a set of $s-1$ variables. If the set of variables represents f, perform this replacement. In this section, we show the s-Min method.

A. Algorithm

For simplicity, we consider only for the cases of $s=2$ and $s=3$.

Algorithm 4.1: (2-Min)

1) Let X_1 be a set of variables that represents f.
2) Select a pair of variables in X_1, and let it be $\{x_i, x_j\}$. Perform the following operations while the number of variables can be reduced.
3) Let $X_2 = X_1 - \{x_i, x_j\}$. When $CD(f : X_2) > 2$, discard this pair.
4) Let $X_3 = X_2 \cup \{y\}$, where $y = x_i \oplus x_j$. If $CD(f : X_3) = 1$, then f can be represented as a function of X_3.

Algorithm 4.2: (3-Min)

1) Let X_1 be a set of variables that represents f.
2) Select a triple of variables in X_1, and let it be $\{x_i, x_j, x_k\}$. Perform the following operations while the number of variables can be reduced.
3) Let $X_2 = X_1 - \{x_i, x_j, x_k\}$. When $CD(f : X_2) > 4$, discard this triple.
4) Let $X_3 = X_2 \cup Y_2$, where Y_2 denotes a pair of compound variables generated by $\{x_i, x_j, x_k\}$. If $CD(f : X_3) = 1$, then X_3 represents f.

B. Amount of Memory

Since Algorithms 4.1 and 4.2 use registered vector tables as a data structure, the necessary memory size is $O(nk)$.

C. Computation Time

The total number combinations to select s variables out of n variables is $\binom{n}{s}$. In the computation of the collision degrees, the register vector table is sorted. Using quick sort, the average time to sort k object is $k \log_2 k$. Thus, the computation time is proportional to $k \log_2 k$[1]. Let s be a small constant (*i.e.*, $s=2$ or $s=3$). Recall that $\binom{n}{s} = \frac{n(n-1)}{2}$ when $s=2$ and $\binom{n}{s} = \frac{n(n-1)(n-2)}{6}$ when $s=3$. Also, we assume that the covering problem can be solved in time proportional to $k \log_2 k$, for each combination, since the covering can be found among a fixed number of combinations. Thus, the total computation time is $O(n^s k \log k)$.

[1]Here, we assume that each object is represented by one word in the computer.

V. Examples

This section illustrates algorithms for 2-Min and 3-Min using examples.

Example 5.1: Consider the 1-out-of-8 code to index converter shown in Table 5.1. By using variables of compound degree 2, we can represent the function with only 5 variables [6]. The compound variables are:

$$\begin{aligned} y_1 &= x_2 \oplus x_6,\ y_2 = x_3 \oplus x_6, \\ y_3 &= x_3 \oplus x_7,\ y_4 = x_4 \oplus x_5, \\ y_5 &= x_5 \oplus x_8. \end{aligned}$$

We have the index generation function $g(y_1, y_2, \ldots, y_5)$ shown in Table 5.2. Note that variables $\{y_1, y_2, y_3, y_4, y_5\}$ distinguish 8 vectors. Now, we apply Algorithm 4.1. From $\{y_1, y_2, y_3, y_4, y_5\}$, we remove $\{y_2, y_5\}$. Then, the remaining variables are, $\vec{Y}_1 = (y_1, y_3, y_4)$. In this case

$$CD(f : Y_1) = Max\{|\{1,8\}|, |\{2,6\}|, |\{3,7\}|, |\{4,5\}|\} = 2.$$

Table 5.1
1-out-of-8 function in Example 5.1.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	f
1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	2
0	0	1	0	0	0	0	0	3
0	0	0	1	0	0	0	0	4
0	0	0	0	1	0	0	0	5
0	0	0	0	0	1	0	0	6
0	0	0	0	0	0	1	0	7
0	0	0	0	0	0	0	1	8

Table 5.2
Function in Example 5.1 represented with variables of compound degree 2.

y_1	y_2	y_3	y_4	y_5	g
0	0	0	0	0	1
1	0	0	0	0	2
0	1	1	0	0	3
0	0	0	1	0	4
0	0	0	1	1	5
1	1	0	0	0	6
0	0	1	0	0	7
0	0	0	0	1	8

Table 5.3
Function in Example 5.1 represented by a variable of compound degree 4.

y_1	z_2	y_3	y_4	h
0	0	0	0	1
1	0	0	0	2
0	1	1	0	3
0	0	0	1	4
0	1	0	1	5
1	1	0	0	6
0	0	1	0	7
0	1	0	0	8

Table 5.4
1-OUT-OF-10 FUNCTION IN EXAMPLE 5.2.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	f
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	2
0	0	1	0	0	0	0	0	0	0	3
0	0	0	1	0	0	0	0	0	0	4
0	0	0	0	1	0	0	0	0	0	5
0	0	0	0	0	1	0	0	0	0	6
0	0	0	0	0	0	1	0	0	0	7
0	0	0	0	0	0	0	1	0	0	8
0	0	0	0	0	0	0	0	1	0	9
0	0	0	0	0	0	0	0	0	1	10

Table 5.5
FUNCTION IN EXAMPLE 5.2 REPRESENTED BY VARIABLES OF COMPOUND DEGREE 2.

y_1	y_2	y_3	y_4	y_5	y_6	g
1	0	0	0	0	0	1
0	0	0	0	0	0	2
0	1	1	0	0	0	3
0	0	0	1	1	0	4
0	0	0	0	0	1	5
1	0	0	0	0	1	6
0	1	0	0	0	0	7
0	0	0	1	0	0	8
0	0	1	0	0	0	9
0	0	0	0	1	0	10

Thus, it may be possible to reduce the number of variables. Let $\vec{Y}_2 = (y_1, z_2, y_3, y_4)$, where

$$z_2 = y_2 \oplus y_5 = x_3 \oplus x_5 \oplus x_6 \oplus x_8.$$

In this case, we have $CD(f : Y_2) = 1$. This shows that the function can be represented with only 4 variables. However, after this, we cannot further reduce the number of variables by Algorithm 4.1 . ■

Example 5.2: Consider the 1-out-of-10 to index converter shown in Table 5.4. By using variables with compound degree 2, we can represent the function with only 6 variables [6]. The compound variables are

$$\begin{aligned} y_1 &= x_1 \oplus x_6,\ y_2 = x_3 \oplus x_7, \\ y_3 &= x_3 \oplus x_9,\ y_4 = x_4 \oplus x_8, \\ y_5 &= x_4 \oplus x_{10},\ y_6 = x_5 \oplus x_6. \end{aligned}$$

We have the index generation function $g(y_1, y_2, \ldots, y_6)$ shown in Table 5.5. Note that 10 vectors can be distinguished with 6 variables.

Table 5.6
FUNCTION IN EXAMPLE 5.2 REPRESENTED WITH VARIABLES OF COMPOUND DEGREE 4.

y_1	y_2	z_3	y_4	z_5	h
1	0	0	0	0	1
0	0	0	0	0	2
0	1	1	0	1	3
0	0	1	1	0	4
0	0	0	0	1	5
1	0	0	0	1	6
0	1	0	0	0	7
0	0	0	1	0	8
0	0	1	0	1	9
0	0	1	0	0	10

Table 6.1
NUMBER OF VARIABLES TO REPRESENT M-OUT-OF-16 FUNCTIONS.

Function	Compound degree			
	$t = 1$	$t = 2$	$t = 3$	$t = 4$
$m = 1$	**15**	11	8	6
$m = 2$	**15**	12	9	8
$m = 3$	**15**	14	11	10
$m = 4$	**15**	14	13	13

First, we try to apply Algorithm 4.1. If we remove any pair of variables from $\{y_1, y_2, y_3, \ldots, y_6\}$, then the collision degree will be 3 or greater. Thus, Algorithm 4.1 cannot be applied. Next, we try to apply Algorithm 4.2. If we remove three variables $\{y_3, y_5, y_6\}$, then the collision degree will be 4. That is, let $\vec{Y}_1 = (y_1, y_2, y_4)$, then we have $CD(f : Y_1) = 4$. We have a chance to reduce the number of variables. Next, consider $\vec{Y}_2 = (y_1, y_2, z_3, y_4, z_5)$, where

$$\begin{aligned} z_3 &= y_3 \oplus y_5 = x_3 \oplus x_9 \oplus x_4 \oplus x_{10}, \\ z_5 &= y_3 \oplus y_6 = x_3 \oplus x_9 \oplus x_5 \oplus x_6. \end{aligned}$$

In this case, we have the index generation function shown in Table 5.6. Since $CD(f : Y_2) = 1$, f can be represented with only 5 variables. ■

VI. EXPERIMENTAL RESULTS

A. *m-out-of-n Code to Index Converters*

Table 6.1 shows the number of variables needed to represent m-out-of-16 functions. These values are taken from [6]. Bold numbers denote minimum values. When only the primitive variables are used ($t = 1$), all functions shown require 15 variables. However, with the increase of the compound degree t, the functions can be represented with fewer variables.

Table 6.2 shows the number of variables when Algorithm 4.1 (2-Min), and Algorithm 4.2 (3-Min) were applied. To obtain these results, we first represent the function by variables with compound degree two ($t = 2$), and then applied 2-Min or 3-Min. Except for the case of $m = 4$, 3-Min reduced more variables than 2-Min. When the functions were represented with only primitive variables ($t = 1$), we could not apply 2-Min.

Similarly, Table 6.3 shows the number of variables to represent m-out-of-20 functions [6]. When only primitive variables were used ($t = 1$), the functions required 19 variables. However, with the increase of compound degree t, functions can be represented with fewer variables.

Table 6.4 shows the number of variables to represent the function when 2-Min and 3-Min were used. To obtain these results, we first represent the function by variables with compound degree two ($t = 2$), and then applied 2-Min or 3-Min. In two of the five cases, 3-Min yielded fewer variables than 2-Min. Again, when the functions were represented with only primitive variables ($t = 1$), we could apply neither 2-Min nor 3-Min.

Table 6.2
NUMBER OF VARIABLES TO REPRESENT M-OUT-OF-16 FUNCTION REDUCED BY 2-MIN AND 3-MIN.

Function	2-Min	3-Min
$m = 1$	8	5
$m = 2$	12	11
$m = 3$	13	12
$m = 4$	14	14

Table 6.3
NUMBERS OF VARIABLES TO REPRESENT M-OUT-OF-20 FUNCTIONS.

Function	Compound degree			
	$t = 1$	$t = 2$	$t = 3$	$t = 4$
$m = 1$	**19**	14	10	8
$m = 2$	**19**	15	12	11
$m = 3$	**19**	17	14	12
$m = 4$	**19**	17	15	15
$m = 5$	**19**	18	17	17

B. Lists of English Words

To compress English text, we can use a list of frequently used words. We made three lists of English words: *List1730*, *List3366*, and *List4705*. The maximum number of characters in the word lists is 13, but we only consider the first 8 characters. For English words consisting of fewer than 8 letters, we append blanks to make the word length 8. We represent each alphabetic character by 5 bits. So, in the lists, all the words are represented by 40 bits. List1730, List3363, and List4705 contain 1730, 3366, and 4705 words, respectively. Within each word list, each English word has a unique index, an integer from 1 to k, where $k = 1730$ or 3360 or 4705. The number of bits for the indices are 11, 12, and 13, respectively. Table 6.5 shows number of variables to represent the list. These results are taken from [6].

Table 6.6 shows the results using 2-Min, and 3-Min. It shows 3-Min obtained better solutions than 2-Min.

C. IP Address Tables

In this experiment, we used distinct IP addresses of computers that accessed our web site over a period of one month. List1670, List3288, List4591, and List7903 contain 1670, 3288, 4591, and 7903 IP addresses, respectively. Table 6.7 shows the results, which are taken from [6]. Note that the original number of variables is 32. The first column shows the function names. The second column shows the number of registered vectors: k. The third column shows the number of variables to represent the function, when only the primitive variables are used (*i.e.* $t = 1$). The fourth column shows the number of variables to represent the function, when the variables with compound degrees up to two are used. Other columns show the number of variables for different values of t. As shown in Table 6.7, the memory size can be reduced when we use compound variables with $t = 3$ or $t = 4$. These results were obtained by the algorithm in [6]. Table 6.8 shows the results using 2-Min, and 3-Min. In these cases, 2-Min and 3-Min could not improve the results of $t = 2$, since the initial solutions for 2-Min and 3-Min were results of $t = 2$. This shows that the qualities of solutions for 2-Min and 3-Min are not so good as that of [6]. But, the computation times are much shorter than that of [6].

Table 6.4
NUMBER OF VARIABLES TO REPRESENT M-OUT-OF-20 FUNCTIONS THAT WERE OBTAINED BY 2-MIN AND 3-MIN.

Function	2-Min	3-Min
$m = 1$	10	5
$m = 2$	14	14
$m = 3$	15	15
$m = 4$	17	16
$m = 5$	18	18

Table 6.5
NUMBER OF VARIABLES TO REPRESENT LISTS OF ENGLISH WORDS

Function		Compound Degree:t			
$Name$	k	1	2	3	4
$List1730$	1730	**31**	19	17	16
$List3366$	3366	**31**	21	19	17
$List4705$	4705	**37**	24	20	19

Table 6.6
NUMBER OF VARIABLES TO REPRESENT LISTS OF ENGLISH WORDS THAT WERE OBTAINED BY 2-MIN AND 3-MIN.

Function	2-Min	3-Min
$List1730$	18	17
$List3366$	20	19
$List4705$	21	20

Table 6.7
NUMBER OF VARIABLES TO REPRESENT IP ADDRESS TABLE.

Function		Compound Degree:t					
$Name$	k	1	2	3	4	5	6
$IP1670$	1670	**18**	17	16	16	15	15
$IP3288$	3288	**20**	19	18	17	17	17
$IP4591$	4591	**21**	20	19	18	18	18
$IP7903$	7903	**23**	21	20	20	20	20

VII. COMPARISON WITH OTHER METHODS

Various methods exist to reduce the number of variables for incompletely specified index generation functions using linear decompositions [4], [5], [6], [7]. Since the number of compound variables to consider is $2^n - 1$, to obtain an exact minimum solution is difficult when n is large. In fact, we have to solve a minimum row cover problem for a table with $O(2^n)$ rows, and $O(k^2)$ columns [7].

Also, to implement the linear part in Fig. 1.1, the circuit cost is low when the compound degree is small. In address

Table 6.8
NUMBER OF VARIABLES TO REPRESENT IP ADDRESS TABLE THAT WERE OBTAINED BY 2-MIN AND 3-MIN.

Function	2-Min	3-Min
$IP1670$	17	17
$IP3288$	19	19
$IP4591$	20	20
$IP7903$	21	21

Table 7.1
COMPARISON WITH EXISTING METHODS

	Exhaustive method [4] ISMVL2011	Heuristic method [6] ASPDAC2012	Heuristic method [7] IEICE2014	Iterative Improvement [5] RM2011	Iterative Improvement This paper
Amount of Memory	$O(k^2 2^n)$	$O(kn^t)$	$O(nk^2)$	$O(nk)$	$O(nk)$
Computation Time	Too Long	$O(n^t k \log_2 k)$	Short	$O(n^5 k)$	$O(n^s k \log_2 k)$
Quality of Solutions	Optimal	Fairly Good	Locally Optimal	Locally Optimal	Locally Optimal

tables of the internet, in many cases, variables with compound degree 2 are sufficient [6].

As for heuristic methods to obtain the compound variables, we have three different methods:

1) Applying the information gain method [2] or reducing ambiguity [6] to the registered vector table.
2) Obtaining the minimum cover of the difference matrix [7], [8].
3) Applying an iterative improvement method [5] to the registered vector table.

In 1), first, we generate all the variables up to compound degree t. The necessary amount of memory is proportional to

$$k\left[\sum_{i=1}^{t}\binom{n}{i}\right],$$

where t is the maximum value of compound degree, k is the number of registered vectors, and n is the number of variables before reduction.

In 2), the necessary amount of memory is proportional to

$$n\binom{k}{2} = n\frac{k(k-1)}{2}.$$

In 1) and 2), when the values of n or k are large, the necessary amount of memory or computation time becomes too large.

In 3), the method [5] uses memory of size $O(nk)$. To find good linear transformations, seven different transformation rules are used to reduce the number of variables. TYPE1 transformation replaces a pair of variables with another pair of variables. TYPE2 and TYPE3 transformations replace a triple of variables with another triple of variables. TYPE4 to TYPE7 transformations replace a quadruple of variables with another quadruple of variables. However, in the method [5], to find the best order of rules to apply is difficult. On the other hand, the s-Min method uses only one rule, and the algorithm is simple.

VIII. SUMMARY

In this paper, we

1) proposed an iterative improvement method (s-Min method) that replace s variables with $(s-1)$ variables.
2) showed that the amount of memory to reduce the number of variables of an n-variable index generation function with weight k is $O(nk)$. Also we showed that the computation time is $O(n^s k \log_2 k)$, when $s = 2$ or $s = 3$.
3) developed computer programs for $s = 2$ and $s = 3$, and showed experimental results.

ACKNOWLEDGMENTS

This research is supported in part by the Grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS). Prof. Jon T. Butler's comments improved English presentation.

REFERENCES

[1] T. Sasao, "On the number of variables to represent sparse logic functions,"*ICCAD-2008*, San Jose, California, USA, Nov. 10-13, 2008, pp. 45-51.

[2] T. Sasao, T. Nakamura, and M. Matsuura, "Representation of incompletely specified index generation functions using minimal number of compound variables," *12th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools* (DSD 2009), Patras, Greece, Aug. 27-29, 2009, pp. 765-772.

[3] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[4] T. Sasao, "Index generation functions: Recent developments,"(invited paper), *International Symposium on Multiple-Valued Logic* (ISMVL-2011), Tuusula, Finland, May 23-25, 2011, pp.1-9.

[5] T. Sasao, "Linear transformations for variable reduction," *Reed-Muller 2011 Workshop*, Tuusula, Finland, May 25-26, 2011.

[6] T. Sasao, "Linear decomposition of index generation functions," *17th Asia and South Pacific Design Automation Conference* (ASPDAC-2012), Jan. 30- Feb. 2, 2012, Sydney, Australia, pp. 781-788.

[7] T. Sasao, Y. Urano, and Y. Iguchi, "A method to find linear decompositions for incompletely specified index generation functions using difference matrix," *IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences*,Vol. E-97A, No. 12, Dec. 2014.

[8] D. A. Simovici, M. Zimand, and D. Pletea,"Several remarks on index generation functions,"*International Symposium on Multiple-Valued Logic* (ISMVL-2012), Victoria, Canada, May 2012, pp. 179-184.