Iterative Linear Transformation to Reduce Compound Variables

Tsutomu Sasao
Department of Computer Science, Meiji University, Kawasaki, Japan

Abstract– A classification function is a multi-valued function, where the function values for only a fraction of the input combinations are defined. Many variables in such a function are redundant, and can be eliminated. A variable that can be represented as an EXOR of variables is called a compound variable. Using compound variables, we can further reduce the number of variables. This paper shows iterative methods to reduce the number of variables. It requires memory with the size $O(nk)$, where n is the number of input variables, and k is the number of the registered vectors. Experimental results for various benchmark functions show the effectiveness of the algorithms. These methods are useful for embedded system, where the memory size is limited. Also, they can be used as a pre-processor for other variable minimizers to reduce computation time.

I. Introduction

A classification function is a multi-valued function, where the function values for only a fraction of the input combinations are defined. In such a function, many variables are redundant, and can be eliminated. A variable that can be represented as an EXOR of variables is called a **compound variable**. By using compound variables, the number of variables to represent a classification function can be further reduced. Fig. 1.1 [3] shows a circuit to represent a classification function, where L realizes linear functions i.e., compound variables, and G realizes general functions. To reduce the number of compound variables, we must find the best linear transformation. Various algorithms [5] have been developed to solve this problem. In general, to obtain a good solution, we need large memory and much CPU time.

In this paper, we consider minimization algorithms of variables in classification functions, which are suitable for an embedded system, where the memory and computation time is limited. We introduce an algorithm called **s-MIN** that reduces the number of variables by iteratively applying simple linear transformations.

The rest of the paper is organized as follows: Section II shows the definitions and basic properties of classification functions and linear decomposition; Section III introduces collision degree and shows its properties; Section IV shows an iterative method for linear transformations to reduce the number of compound variables; Section V shows an example illustrating the algorithm; SectionVI reviews existing methods to reduce the number of compound variables; Section VII shows experimental results; Section VIII shows a method to use the s-MIN method to reduce CPU time for other minimizers, and finally Section IX concludes the paper.

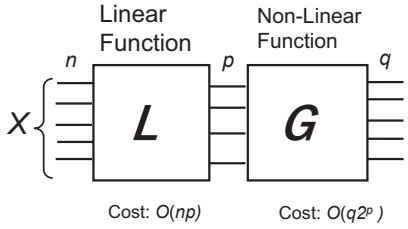

Fig. 1.1. Linear decomposition.

TABLE 2.1
Registered Vector Table for f

x_1	x_2	x_3	x_4	x_5	x_6	f
1	1	0	0	1	1	0
0	1	1	0	1	1	0
0	1	0	1	0	0	0
0	0	0	0	1	0	0
1	1	0	1	1	1	1
1	0	1	1	1	1	1
1	0	0	0	1	1	1
0	0	1	0	1	0	1

II. Definitions and Basic Properties

Definition 2.1 *Let n be the number of variables, $B = \{0,1\}$, $D \subset B^n$, m be the number of classes, and $M = \{0,1,2,\ldots,m-1\}$. Then, a* **classification function** *is a mapping: $f : D \to M$.*

Example 2.1 Table 2.1 is a **registered vector table**. It shows the function with $n = 6$, $m = 2$, and $|D| = 8$. Each vector in the table is a **registered vector**. There are 2^6 possible input combinations, but the function is defined for only 8 combinations. For the other $64 - 8 = 56$ combinations, function values are undefined. ■

Next, we show a simple method to reduce the number of variables in classification functions.

Example 2.2 The function f in Table 2.1 can be represented by x_1, x_2, x_3 and x_4. That is, four variables are sufficient to

TABLE 2.2
f CAN BE REPRESENTED WITH x_1, x_2, x_3 AND x_4.

x_1	x_2	x_3	x_4	x_5	x_6	f
1	1	0	0	1	1	0
0	1	1	0	1	1	0
0	1	0	1	0	0	0
0	0	0	0	1	0	0
1	1	0	1	1	1	1
1	0	1	1	1	1	1
1	0	0	0	1	1	1
0	0	1	0	1	0	1

TABLE 2.3
f CANNOT BE REPRESENTED WITH x_3, x_4, x_5 AND x_6.

x_1	x_2	x_3	x_4	x_5	x_6	f	
1	1	0	0	1	1	0	*
0	1	1	0	1	1	0	
0	1	0	1	0	0	0	
0	0	0	0	1	0	0	
1	1	0	1	1	1	1	
1	0	1	1	1	1	1	
1	0	0	0	1	1	1	*
0	0	1	0	1	0	1	

represent the function. As shown in Table 2.2, the first four bits uniquely specify the function values.

However, the same function cannot be represented with x_3, x_4, x_5 and x_6. Note that when $(x_3, x_4, x_5, x_6) = (0, 0, 1, 1)$, the value of f is not unique. In Table 2.3, the rows with asterisks show inconsistency. ■

Consider the decomposition of the function shown in Fig. 1.1, where L contains a linear function, while G contains a general function. The cost of L is $O(np)$, while the cost of G is $O(q2^p)$. When n is large, the cost of L can be neglected. The functions produced by the circuit L in Fig. 1.1 have the form:

$$y_i = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n,$$

where $a_j \in \{0, 1\}$. y_i is called a **compound variable**. $\sum_{i=1}^{n} a_i$ is the **compound degree**. When the compound degree is one, y_i is called a **primitive variable**.

III. COLLISION DEGREE AND ITS PROPERTIES

In this section, we introduce **collision degree** to find a good linear transformation.

Definition 3.1 *Let $f(X)$ be a classification function, where $X = \{x_1, x_2, \ldots, x_n\}$ is the set of variables in f. Let $X_1 \subset X$. 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 set to $\vec{a} = (a_1, a_2, \ldots, a_s)$, where $a_i \in B$. Let $N(f, \vec{X}_1, \vec{a})$ be the number of different specified values of f. 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 in B^s denotes the number of variables in X_1.

TABLE 3.1
CLASSIFICATION FUNCTION f.

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

TABLE 3.2
CLASSIFICATION FUNCTION g.

y_1	x_2	x_3	g
1	0	0	1
0	1	0	2
1	0	1	1
0	0	0	3

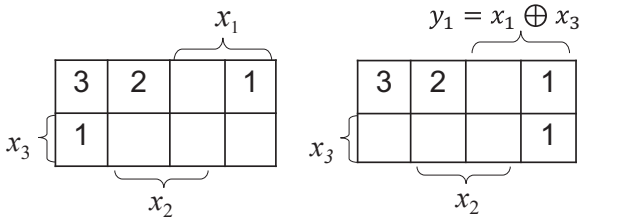

Fig. 3.1. Maps for Classification Function.

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

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

Lemma 3.1 *Consider the decomposition chart of $f(X)$, where X_1 specifies the variables labeling the columns and $X - X_1$ denotes the variables labeling the rows. Then, $N(f, \vec{X}_1, \vec{a})$ shows the number of different specified values in a column, and the collision degree $CD(f : X_1)$ is the maximal number of different values in the columns.*

Example 3.2 The map in the left-hand side of Fig. 3.1 can be considered as the decomposition chart of the classification function shown in Table 3.1. In this chart, the column variables are $X_1 = \{x_1, x_2\}$, and blank elements show *don't cares*. The number of different specified values in each column is, from the left to the right, 2, 1, 0, 1. Note that the leftmost column has the maximum number of different values, 2. Thus, $CD(f : \{x_1, x_2\}) = 2$. ■

A classification function $f(X)$ can be represented by a subset X_1 of X if every assignment of values to the variables X_1 uniquely specifies the value of f.

TABLE 3.3
CLASSIFICATION FUNCTION f.

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

TABLE 3.4
CLASSIFICATION FUNCTION g.

y_1	y_2	x_3	g
0	0	0	3
1	0	0	1
0	1	0	2
1	1	1	1
0	0	1	3

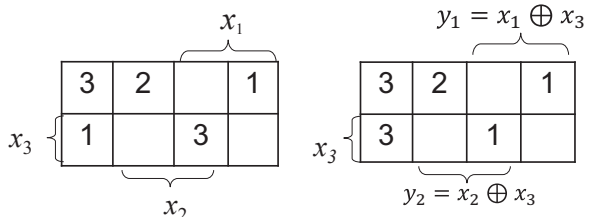

Fig. 3.2. Maps for Classification Function.

Theorem 3.1 *Let $f(X)$ be a classification function, and X_1 be a proper subset of X. Then, f can be represented as a function of X_1, if*

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

Example 3.3 Consider the classification function g shown in Table 3.2. This table is derived from Table 3.1 by replacing x_1 with $y_1 = x_1 \oplus x_3$. Since $CD(g : \{y_1, x_2\}) = 1$, g can be represented with only y_1 and x_2. The map in the right-hand side of Fig. 3.1 shows g. For example, g can be represented as

$$\mathcal{G} = 1 \cdot y_1 \vee 2 \cdot \bar{y}_1 x_2 \vee 3 \cdot \bar{y}_1 \bar{x}_2,$$

where the symbol $\vee$ denotes the max operator, and $f(x_1, x_2, x_3) = g(y_1, x_2)$.

However, f in Table 3.1 requires three variables to represent the function:

$$\mathcal{F} = 1 \cdot x_1 \bar{x}_2 \bar{x}_3 \vee 2 \cdot \bar{x}_1 x_2 \bar{x}_3 \vee 3 \cdot \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee 1 \cdot \bar{x}_1 \bar{x}_2 x_3.$$

Note that the minterm in the lowest leftmost position of Fig. 3.1 is moved to the rightmost position by the linear transformation: $y_1 = x_1 \oplus x_3$. ■

Example 3.4 Consider the classification function f shown in Table 3.3. In this case, f requires three variables. Here, we consider the linear transformation:

$$y_1 = x_1 \oplus x_3, \qquad y_2 = x_2 \oplus x_3.$$

Table 3.4 shows the transformed function g.

The original function f is shown by the map in the left-hand side of Fig. 3.2, while the transformed function g is shown in the map in the right-hand side of Fig. 3.2, Since $CD(g : \{y_1, y_2\}) = 1$, the function can be represented with only y_1 and y_2. In fact, the function can be represented as

$$\mathcal{G} = 1 \cdot y_1 \vee 2 \cdot \bar{y}_1 y_2 \vee 3 \cdot \bar{y}_1 \bar{y}_2,$$

where $f(x_1, x_2, x_3) = g(y_1, y_2)$. Note that, for this function, the previous transformation $y_1 = x_1 \oplus x_3$ cannot reduce the number of variables. ■

Theorem 3.2 *Let $f(X)$ be a classification 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.*

Corollary 3.1 *Let $f(X)$ be a classification 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 a classification 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

To find a linear transformation that minimizes the number of compound variables p in a classification function, is very difficult. Various heuristic methods are known [5]. Most methods require a large amount of memory and CPU time.

Here, we use the **s-MIN method**[1], where a set of s variables in X_1 is replaced with a set of $s - 1$ compound variables. If the resulting 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 with $s = 2$ and $s = 3$.

Algorithm 4.1 (2-MIN)

1. Let X be a set of variables on which f depends.
2. For all possible pairs of variables in X, $\{x_i, x_j\}$, perform the following operations while the number of variables can be reduced.
3. Let $X_1 = X \setminus \{x_i, x_j\}$. When $CD(f : X_1) > 2$, discard this pair.
4. Let $X_3 = X_1 \cup \{y_1\}$, where $y_1 = x_i \oplus x_j$. If $CD(g : X_3) = 1$, then g can be represented as $g(X_3) = f(X)$, and Stop. Otherwise, discard this pair.

Algorithm 4.2 (3-MIN)

1. Let X be a set of variables on which f depends.

[1]This method was originally developed for index generation functions [4]. We found this method is also effective for classification functions.

2. For all possible triples of variables in X, $\{x_i, x_j, x_k\}$, perform the following operations while the number of variables can be reduced.

3. Let $X_1 = X \setminus \{x_i, x_j, x_k\}$. When $CD(f : X_1) > 4$, discard this triple.

4. Let $X_3 = X_1 \cup \{y_i, y_j\}$, where

$$y_i = x_i \oplus x_k, \qquad y_j = x_j \oplus x_k.$$

If $CD(g : X_3) = 1$, then g can be represented as $g(X_3) = f(X)$, and Stop. Otherwise discard this triple.

Note that the solutions produced by Algorithms 4.1 and 4.2 are local optimal.

B. Amount of Memory and Computation Time

Since Algorithms 4.1 and 4.2 use registered vector tables as a data structure, the necessary memory size is $O(nk)$, where n is the number of variables, and k is the number of registered vectors.

The total number of combinations to select s variables out of n variables is $\binom{n}{s}$. In the computation of the collision degrees, the registered vectors are sorted. Using quick sort, the average time to sort k objects is $k \log_2 k$. Thus, the CPU time[2] is proportional to $k \log_2 k$. Let s be a small constant (*i.e.*, $s = 2$ or $s = 3$). Recall that $\binom{n}{2} = \frac{n(n-1)}{2}$, and $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$. Thus, the total CPU time is $O(n^s k \log k)$.

V. Examples

This section illustrates the algorithm for 2-MIN.

TABLE 5.1
ORIGINAL FUNCTION.

x_1	x_2	x_3	x_4	x_5	f
0	1	1	1	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	1	0	0	1
1	1	1	0	1	1
0	0	0	0	1	2
0	0	0	1	0	2
0	0	1	0	0	2
0	0	1	1	1	2
0	1	0	0	1	3
1	0	1	1	1	3
1	1	0	0	1	3
1	1	0	1	0	3

TABLE 5.2
AFTER LINEAR TRANSFORM.

x_1	x_2	x_3	y	g
0	1	1	1	1
1	0	1	1	1
1	0	1	1	1
1	1	1	0	1
1	1	1	1	1
0	0	0	1	2
0	0	0	1	2
0	0	1	0	2
0	0	1	0	2
0	1	0	1	3
1	0	1	0	3
1	1	0	1	3
1	1	0	1	3

Example 5.1 Consider the 5-variable classification function shown in Table 5.1. Note that no variable is redundant in the table. Let $X = (x_1, x_2, x_3, x_4, x_5)$. Let the pair of variables be $\{x_4, x_5\}$. $X_1 = X \setminus \{x_4, x_5\}$. Note that when $(x_1, x_2, x_3) = (1, 0, 1)$, f can take two different values 1 or 3. For other combinations, f takes a unique value. Thus, $CD(f : X_1) = 2$. By Corollary 3.1, there is a possibility to reduce the variables. Let $X_3 = X_1 \cup \{y\}$, where $y = x_4 \oplus x_5$. Note that $CD(g : X_3) = 1$. Thus, g can be represented by X_3 as shown in Table 5.2. However, after this, we cannot reduce the number of variables.

Note that in Table 5.1, all the registered vectors are distinct. However, in Table 5.2, identical registered vectors appear. For example, $(x_1, x_2, x_3, y) = (0, 0, 0, 1)$ appears twice. Thus, the linear transformation not only reduces the number of variables, but also reduces the number of distinct registered vectors. ■

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

VI. Review of Existing Methods

Various methods to reduce the number of (compound) variables in classification functions have been developed [5]. In this part, we briefly review three methods.

A. Minimization of Primitive Variables

This method reduces the number of primitive variables. **Impurity measure** [5] is used to select important variables. A reduction of variables corresponds to reduce the depth of the classification tree [1]. The impurity measure is used to order the variables in the classification tree.

Algorithm 6.1 (A reduction of primitive variables)
Given a classification table of a classification function f.

1. Compute the impurity measure $\mu(\vec{e}_i)$ for $i = 1, 2, \ldots, n$. Note that $\vec{e}_i$ denotes the unit vector, where only the i-th element is 1, and other elements are 0s.

2. Assume that $\mu(\vec{e}_i)$ is the minimum. Let $\vec{a} \leftarrow e_i$. $\vec{a}$ shows the set of selected variables.

3. Select a variable x_j from the remaining set of variables, so that the measure is minimized for the resulting classification tree. Let $\vec{a} \leftarrow \vec{a} \vee \vec{e}_j$.

4. If $\mu(\vec{a}) > 0$, then go to step 3, else stop.

The method is very fast. The memory size for the algorithm is $O(nk)$, where n is the number of input variables, and k is the number of registered vectors.

B. Minimization of Compound Variables with Degree Two

This algorithm first generates all the compound variables with degree two, and then selects necessary variables using the impurity measure. The number of compound variables is $n(n-1)/2$. The size of memory for the algorithm is $O(n^2 k)$.

Algorithm 6.2 This algorithm is similar to Algorithm 6.1. In the registered vector table, append the compound variables with degree two. Thus, the total number of variables is $n + \binom{n}{2} = \frac{(n+1)n}{2}$.

C. Iterative Linear Transformation using Difference Vectors

This algorithm first generates the set of difference vectors D_f, and then select linear transformations using D_f, iteratively.

Definition 6.1 *In a classification function f, let $f(\vec{a}) \neq f(\vec{b})$, where $\vec{a}, \vec{b} \in D$, $\vec{a} \neq \vec{b}$, and D is the set of registered vectors. Then, the vector $\vec{d} = \vec{a} \oplus \vec{b}$ is a* **difference vector***, where $\oplus$ denotes the bitwise EXOR operator. The set of all difference vectors is denoted by D_f.*

Lemma 6.1 *Let $f : D \rightarrow M$, $M = \{0, 1, \ldots, m-1\}$ be a classification function. Let $|D_f|$ be the number of distinct difference vectors. Then,*

$$|D_f| \leq \sum_{(i<j)} k_i k_j,$$

where $i, j \in \{1, 2, \ldots, m\}$, and k_i is the number of vectors $\vec{a} \in D$, such that $f(\vec{a}) = i$.

Algorithm 6.3 (Reduction of Compound Variables)

1. Derive the set of difference vectors D_f of an n-variable function.
2. If $|D_f| = 2^n - 1$, then stop, since reduction is impossible.
3. Obtain a non-zero vector $\vec{d} \in B^n \setminus D_f$ with the minimum weight[3].
4. Remove one variable from $\vec{d}$, and apply the linear transformation to the function.
5. Let $n \leftarrow n - 1$, and go to step 1. If $n = 1$, Stop.

The number of distinct difference vectors is $O(k^2)$. Thus, the size of memory for the algorithm is $O(nk^2)$. It produces very good solutions, but requires large memory size and much CPU time.

VII. Experimental Results

To investigate the performance of the s-MIN method, we reduced the compound variables for various benchmark functions [5]. Table 7.2 shows the results. The first four column shows the properties of the function. The column headed with *Name* shows the function name; the column headed with n shows the number of variables in the original function; the column headed with m shows the number of classes; and the column headed with k shows the number of the registered vectors. The fifth column headed with ALG1 shows the number of the primitive variables obtained by Algorithm 6.1. The sixth column headed with ALG2 shows the number of the compound variables with degree two obtained by Algorithm 6.2.

The next three columns headed with ALG3 show the result of Algorithm 6.3. The column headed with Vari shows the number of compound variables; the column headed with Max shows the maximum compound degree; and the column headed with AVG shows the average compound degree.

The next three columns show the results for 2-MIN. The last three columns show the results for 3-MIN.

The boldface numbers in the bottom row of Table 7.2 show the total numbers of variables. From these, we can observe the tendency:

$$\text{ALG1} > \text{2-MIN} > \text{ALG2} > \text{3-MIN} > \text{ALG3}.$$

Note that 3-MIN produces solutions with fewer variable than 2-MIN. However, 3-MIN produces compound variables with higher degrees. Note that some entries do not satisfy the above relation.

TABLE 7.1
Complexity of Reduction Algorithms

Algorithm	Degree	CPU Time	Memory
Algorithm 6.1	1	$O(nk \log k)$	$O(nk)$
Algorithm 6.2	2	$O(n^2 k \log k)$	$O(n^2 k)$
Algorithm 6.3	3+	$O(nk^2)$	$O(nk^2)$
2-MIN	3+	$O(n^2 k \log k)$	$O(nk)$
3-MIN	3+	$O(n^3 k \log k)$	$O(nk)$

Table 7.1 compares complexities of minimization algorithms, where n is the number of variables in the original function, and k is the number of registered vectors. Algorithm 6.1 requires the shortest CPU time, while Algorithm 6.3 requires the longest CPU time, when k is large.

VIII. An Application as a Pre-Processor

Table 7.2 shows that Algorithm 6.3 obtains the best solutions. Unfortunately, it is time-consuming for functions with large k, as shown in Table 7.1. In this part, we use 3-MIN to reduce the total CPU time for the MNIST28×28 function for illustration. The function implements a handwritten digits recognition circuit [2]. It has $n_0 = 784$ variables, $k = 59981$ registered vectors, and $m = 10$ classes. Algorithm 6.1 yields $n_1 = 37$-variable solution in 57 seconds. While, Algorithm 6.3 yields an $n_3 = 25$-variable solution in 2718 seconds, which is very time-consuming.

However, we can reduce the CPU time by using 3-MIN as shown in Fig. 7.1. First, we apply Algorithm 6.1 to obtain an

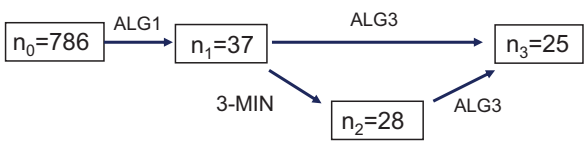

Fig. 7.1. Reduction of CPU time for MNIST 28×28.

[3]The **weight of a vector** is the number of 1's in the vector.

TABLE 7.2
RESULTS OF LINEAR TRANSFORMATIONS

Function Data				Existing Methods					Proposed Methods					
				ALG1	ALG2	ALG3			2-MIN			3-MIN		
Name	n	m	k	Vari	Vari	Vari	Max	AVG	Vari	Max	AVG	Vari	Max	AVG
4350WORDS	75	14	4,350	43	25	18	14	3.94	21	10	2.05	20	16	4.85
CHEMICAL ELEMENTS	60	7	118	11	9	9	3	1.44	11	13	2.55	9	29	5.60
CHESS3196	75	2	3,196	30	21	15	11	3.67	18	6	1.67	16	11	4.06
CIFAR32 × 32	1024	2	9,930	58	30	19	22	7.47	25	10	2.32	22	22	8.05
COMPANIES	30	9	3,700	21	19	18	3	1.39	20	4	1.40	19	9	3.21
CONNECT-4	126	3	67,557	61	38	22	23	11.82	34	6	1.79	28	22	7.43
COUNTRIES	60	6	197	15	10	10	4	2.30	12	15	3.17	11	33	12.91
LETTER-RECOG.	256	26	20,000	51	32	21	17	9.05	36	6	1.42	28	17	5.36
LYMA	59	4	147	11	8	8	3	1.62	9	3	1.22	8	3	1.62
MNIST14 × 14	196	10	58,191	45	29	25	10	4.00	32	5	1.41	28	11	3.46
MNIST28 × 28	784	10	59,981	37	29	25	7	2.88	30	3	1.23	28	4	1.82
MNIST2CLASS	784	2	59,984	35	25	24	9	3.08	31	3	1.13	27	6	2.00
MNIST8 × 8	64	10	3,686	23	19	17	5	2.24	20	10	2.10	19	22	13.05
POKER	85	10	25,010	61	37	21	23	11.95	43	5	1.42	23	23	9.39
RANDOM4000	30	4	4,000	21	20	18	4	1.72	20	2	1.47	19	9	3.11
SPAM MAIL FILTER	128	2	20,000	23	22	22	2	1.36	23	1	1.00	23	1	1.00
SPLICE	240	3	3,174	18	17	15	4	2.00	18	1	1.00	17	2	1.12
Total				**564**	**391**	**307**		71.93	**403**		28.35	**345**		88.04

$n_1 = 37$-variable solution. Second, we apply 3-MIN to obtain an $n_2 = 28$ variable solution in 42 seconds. And, finally we apply Algorithm 6.3 to obtain an $n_3 = 25$-variable solution in 747 seconds. In this way, the CPU time for Algorithm 6.3 can be reduced to less than one third.

The 3-MIN reduced both the number of input variables n, and the number of distinct registered vectors k. This reduced the number of distinct difference vectors. The number of distinct difference vectors in the result of Algorithm 6.1 is 1,226,244,862. However, after applying 3-MIN, the number of distinct difference vectors is reduced to 259,485,042. Also the number of variables is reduced from 37 to 28, and the number of the iterations is reduced from 12 to 3. These account for the reduction of the CPU time.

We used a computer with an INTEL Core i7, 7700, 3.65GHz CPU, and 64 GB main memory, on Windows 10.

IX. CONCLUSION

This paper presented minimization algorithms for (compound) variables to represent classification functions. They require small memory and time complexities.

The main results are as follows:

1. 2-MIN performs a linear transformation with type

$$x_i \leftarrow x_i \oplus x_j$$

iteratively, to reduce the number of variables.

2. 3-MIN performs a linear transformation with type

$$x_i \leftarrow x_i \oplus x_k, \qquad x_j \leftarrow x_j \oplus x_k$$

iteratively, to reduce the number of variables.

3. They are fast and require small memory size.

4. Experimental results using various benchmark functions show that 3-MIN produces solutions with fewer compound variables than 2-MIN, on the average. However, 3-MIN tends to produce solutions with higher compound degrees.

5. These algorithms are suitable for embedded systems and pre-preprocessing for further minimization.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. The author thanks Prof. Jon T. Butler and Dr. Alan Mishchenko for discussion.

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning,* Springer, 2006.

[2] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, " Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, November 1998.

[3] E. I. Nechiporuk, "On the synthesis of networks using linear transformations of variables," *Dokl. AN SSSR*, vol. 123, no. 4, Dec. 1958, pp. 610-612 (in Russian).

[4] T. Sasao, "A reduction method for the number of variables to represent index generation functions: s-Min method," *ISMVL*, May 18-20, 2015, Waterloo, Canada, pp. 164-169.

[5] T. Sasao, *Classification Functions for Machine Learning and Data Mining*, Springer Nature, Aug. 2023.