# Linear Transformations for Iterative Reduction of 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. Using linear transformation, we can further reduce the number of variables. The problem is to find the linear transformation that minimizes the number of variables. This paper considers iterative methods $s$-MIN to reduce the number of variables, by replacing a set of $s$ variables with other set of $s-1$ variables, by linear transformations. 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. The time complexity is $O(n^s k \log k)$. We compare the performance of various linear transformations by experiments. Also, we show a method to use $s$-MIN as a pre-processor for other minimizer to reduce total computation time.**

***Index Terms*—classification, functional decomposition, linear transformation, minimization of variables, MNIST, partially defined function.**

## 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. We assume that $L$ is implemented by EXOR gates, while $G$ is implemented by a memory. When $n$, the number of input variables, is large, the cost for $L$ can be neglected. To reduce $p$, the number of compound variables, we must find the linear transformation that minimizes $p$. In general, to obtain a good linear transformation, we need large memory and much CPU time. Various algorithms [9] have been developed to solve this problem.

In this paper, we consider 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 $s$-MIN, iterative methods for linear transformations to reduce the number of compound variables; Section V analyzes linear transformations using matrices; Section VI reviews existing methods to reduce the number of variables; Section VII shows experimental results; Section VIII shows a method to use $s$-MIN as a preprocessor to reduce total CPU time with other minimizer, and finally Section IX concludes the paper.

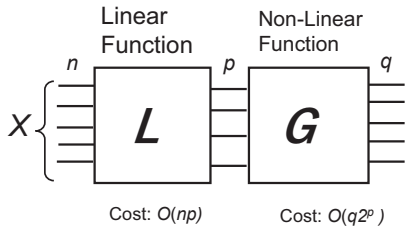


Fig. 1.1. Linear decomposition.

## II. Definitions and Basic Properties

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

**Example 2.1** Table 2.1 is a **registered vector table** of a classification function. It shows the function with $n = 3$, $m = 3$, and $|\mathcal{D}| = 4$. Each vector in the table is a **registered vector**. There are $2^3$ possible input combinations, but the function is defined for only 4 combinations. For the other $8 - 4 = 4$ combinations, function values are undefined. ■

TABLE 2.1
Classification Function $f$.

| $x_1$ | $x_2$ | $x_3$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 3 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 |

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:

$$y_i = a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n,$$

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 efficiently.

TABLE 3.1
CLASSIFICATION FUNCTION $g$.

| $y_1$ | $x_2$ | $x_3$ | $g$ |
|---|---|---|---|
| 0 | 0 | 0 | 3 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 1 |

**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 $a_i \in \mathcal{B}$, and $s$ denotes the number of variables in $X_1$. Let $N(f, \vec{X}_1, \vec{a})$ be the number of different specified values of $f$. Then, the* **collision degree** *is*

$$CD(f : X_1) = \max_{\vec{a} \in \mathcal{B}^s} \left\{ N(f, \vec{X}_1, \vec{a}) \right\}.$$

**Example 3.1** Consider the classification function $f$ shown in Table 2.1. We have:

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

■

**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 the column for $\vec{a}$, 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 2.1. 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, $CD(f : \{x_1, x_2\}) = 2$. ■

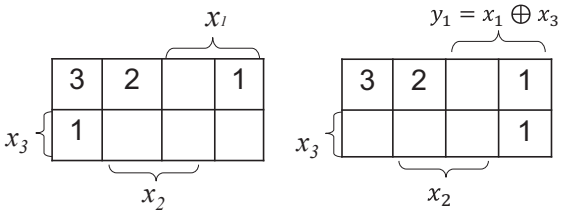


Fig. 3.1. Maps for Classification Function.

**Example 3.3** Consider the classification function $g$ shown in Table 3.1. This table is derived from Table 2.1 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 2.1 requires three variables to represent the function:

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

When $x_3 = 1$, the element in the leftmost column of Fig. 3.1 is moved to the rightmost column by the linear transformation: $y_1 = x_1 \oplus x_3$. ■

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 $f$.

**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 $CD(f : X_1) = 1$.*
- *At least $\lceil \log_2 CD(f : X_1) \rceil$ compound variables are necessary in addition to the variables in $X_1$ to represent $f(X)$.*

**Corollary 3.1** *Let $f(X)$ be a classification function, and let $X_1$ be a proper subset of $X$. A necessary condition that $f$ can be represented by $X_1$ and*

- *one compound variable is $CD(f : X_1) = 2$.*
- *a pair of compound variables is $CD(f : X_1) \leq 4$.*
- *a triple of compound variable is $CD(f : X_1) \leq 8$.*
- *a quadruple of compound variables is $CD(f : X_1) \leq 16$.*

## IV. $s$-MIN METHOD

Finding a linear transformation that minimizes the number of compound variables $p$ in a classification function, is very difficult. Various heuristic methods are known [9]. Most methods require a large amount of memory and long 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 $f$, then we perform this replacement. In this section, we show the $s$-MIN method.

### A. Algorithm

For simplicity, only the cases for $s = 2$ is shown. Algorithms for 3-MIN, 4-MIN, and 5-MIN are similar.

**Algorithm 4.1** (2-MIN)

1) Let $X$ be a set of variables for $f$.

[1]This method was originally developed for index generation functions [7], [8]. We found this method is also effective for classification functions.

2) For all possible pairs $\{x_i, x_j\}$ of variables in $X$, 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 and return.
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 return. Otherwise, discard this pair, and return.

### *B. Examples*

**2-MIN:** Consider the 4-variable functions in Fig. 4.1. The upper chart shows that this function requires 4-variables, since it has a collision in the column of $(x_1, x_2, x_3) = (0, 1, 1)$. However, if we use 2-MIN to perform $y_1 = x_1 \oplus x_4$, the elements in the bottom row are permuted as shown in the lower chart, where the values of $f(x_1, a_2, a_3)$ are swapped with that of $f(\bar{x}_1, a_2, a_3)$ to avoid the collision. Thus, $x_4$ becomes redundant.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $x_1$ $zw$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $x_2$ $zw$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $x_3$ $zw$ |
| 0 | | | | 3 | 1 | | | | |
| $x_4$ 1 | 3 | 4 | | 2 | 2 | 4 | | | |

$\Downarrow$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $y_1 = x_1 \oplus x_4$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $x_2$ $zw$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $x_3$ $zw$ |
| 0 | | | 3 | 1 | | | | | |
| $x_4$ 1 | **2** | **4** | | | **3** | **4** | | **2** | |

Fig. 4.1. Example of 2-MIN

**3-MIN:** Consider the 4-variable functions in Fig. 4.2. The upper chart shows that this function requires 4-variables, since it has a collision in the column of $(x_1, x_2, x_3) = (1, 1, 1)$. However, if we use 3-MIN to perform $y_2 = x_2 \oplus x_4$, and $y_3 = x_3 \oplus x_4$, as shown in the lower chart, the elements in the bottom row are permuted, where the values of $f(a_1, x_2, x_3)$ are swapped with that of $f(a_1, \bar{x}_2, \bar{x}_3)$ to avoid the collision. Thus, $x_4$ becomes redundant. Note that, for this function, 2-MIN cannot reduce the variable.

**4-MIN:** Consider the 4-variable functions in Fig. 4.3. The upper chart shows that this function requires 4-variables, since it has three collisions. However, if we use 4-MIN to perform $y_1 = x_1 \oplus x_4$, $y_2 = x_2 \oplus x_4$, and $y_3 = x_3 \oplus x_4$, as shown in the lower chart, the elements in the bottom row are permuted, where the values of $f(x_1, x_2, x_3)$ are swapped with that of $f(\bar{x}_1, \bar{x}_2, \bar{x}_3)$ to avoid the collisions. Thus, $x_4$ becomes redundant. Note that, for this function, neither 2-MIN nor 3-MIN can reduce the variable.

### *C. Amount of Memory and Computation Time*

Since Algorithm 4.1 uses 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.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $x_1$ $zw$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $x_2$ $zw$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $x_3$ $zw$ |
| 0 | | | | | | | | 4 | |
| $x_4$ 1 | | | | 1 | | 2 | 3 | 5 | |

$\Downarrow$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $x_1$ $zw$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $y_2 = x_2 \oplus x_4$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $y_3 = x_3 \oplus x_4$ |
| 0 | | | | | | | | 4 | |
| $x_4$ 1 | **1** | | | | **5** | **3** | **2** | | |

Fig. 4.2. Example 3-MIN

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $x_1$ $zw$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $x_2$ $zw$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $x_3$ $zw$ |
| 0 | 1 | | | | 2 | 3 | 4 | 5 | |
| $x_4$ 1 | | | | | 6 | 7 | 8 | | |

$\Downarrow$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | $y_1 = x_1 \oplus x_4$ |
| | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | $y_2 = x_2 \oplus x_4$ |
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $y_3 = x_3 \oplus x_4$ |
| 0 | 1 | | | | 2 | 3 | 4 | 5 | |
| $x_4$ 1 | | **8** | **7** | **6** | | | | | |

Fig. 4.3. Example of 4-MIN

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.*, $s = 2, 3, 4, 5$), then the total CPU time is $O(n^s k \log k)$.

## V. Analysis of Linear Transformations

This part analyses linear transformations using matrices. Note that the computation of matrices is done in GF(2).

$\mathbf{n = 2}$ : Linear transformations for $n = 2$ include

$$M_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } M_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

$M_1$ corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_2, \\ x_2 &\Leftarrow x_2. \end{aligned}$$

Note that $M_1 M_1 = (M_1)^2 = I_2$, where $I_2$ is the identity matrix, and $M_2 = (M_1)^T$, i.e., $M_2$ is equivalent to $M_1$ if we consider the permutation of variables $(x_1, x_2)$. Thus, when $n = 2$, the only transformation to consider is $M_1$.

$\mathbf{n = 3}$ : Linear transformations for $n = 3$ include

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

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

$A$ is the simplest non-trivial linear transformation for $n = 3$, and corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_3, \\ x_2 &\Leftarrow x_2 \oplus x_3, \\ x_3 &\Leftarrow x_3. \end{aligned}$$

Note that $A^2 = I_3$, where $I_3$ is the identity matrix.

$B$ is another linear transformation that corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_2 \oplus x_3, \\ x_2 &\Leftarrow x_2 \oplus x_3, \\ .x_3 &\Leftarrow x_3. \end{aligned}$$

Note that $B^4 = I_3$, so $B$ is a generator of a cyclic group of order 4.

$\mathbf{n = 4}$ : Linear transformations for $n = 4$ include

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } D = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$C$ is the simplest non-trivial linear transformation that corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_4, \\ x_2 &\Leftarrow x_2 \oplus x_4, \\ x_3 &\Leftarrow x_3 \oplus x_4, \\ x_4 &\Leftarrow x_4. \end{aligned}$$

Note that $C^2 = I_4$, where $I_4$ is the identity matrix.

$D$ is another linear transformation that corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_2 \oplus x_3 \oplus x_4, \\ x_2 &\Leftarrow x_2 \oplus x_3 \oplus x_4, \\ x_3 &\Leftarrow x_3 \oplus x_4, \\ x_4 &\Leftarrow x_4. \end{aligned}$$

Note that $D^4 = I_4$, so $D$ is a generator of a cyclic group of order 4.

$\mathbf{n = 5}$ : Linear transformations for $n = 5$ include

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } F = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$E$ is the simplest non-trivial linear transformation that corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_5, \\ x_2 &\Leftarrow x_2 \oplus x_5, \\ x_3 &\Leftarrow x_3 \oplus x_5, \\ x_4 &\Leftarrow x_4 \oplus x_5, \\ x_5 &\Leftarrow x_5. \end{aligned}$$

Note that $E^2 = I_5$, where $I_5$ is the identity matrix.

$F$ is another linear transformation that corresponds to

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5, \\ x_2 &\Leftarrow x_2 \oplus x_3 \oplus x_4 \oplus x_5, \\ x_3 &\Leftarrow x_3 \oplus x_4 \oplus x_5, \\ x_4 &\Leftarrow x_4 \oplus x_5, \\ x_5 &\Leftarrow x_5. \end{aligned}$$

Note that $F^8 = I_5$, so $F$ is a generator of a cyclic group of order 8.

To find an efficient minimization algorithm, we have to select the best linear transformations [4]. Since, $B$, $D$, and $F$ can generate more patterns than $A$, $C$, and $E$, respectively, we thought that they might produce better solutions. However, experimental results in Section VII showed that, in many cases, $B$, $D$, and $F$, produced solutions with more variables than $A$, $C$, and $E$, respectively. Thus, for the benchmark functions in Section VII, $A$, $C$, and $E$ are more suitable for $s$-MIN algorithms.

## VI. Review of Existing Methods

Various methods to reduce the number of (compound) variables in classification functions have been developed [9]. In this part, we briefly review three methods [6], [9], [10].

### A. Minimization of Primitive Variables

This method reduces the number of primitive variables without considering linear transformation. For each variable, this method checks if a variable is necessary or not. And try to select necessary variables in a greedy way. Since the order of the selection of variables influence the quality of the results, we use **impurity measure** [9] 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** (ALG1: 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 in a greedy way using the impurity measure. The number of compound variables is $n(n-1)/2$. The size of memory for the algorithm is $O(n^2k)$.

**Algorithm 6.2** (ALG2: Reduction of compound variables with degree two) 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 method first generates the set of difference vectors $\mathcal{D}_f$, and then select linear transformations using $\mathcal{D}_f$, iteratively.

**Definition 6.1** *In a classification function $f$, let $f(\vec{a}) \neq f(\vec{b})$, where $\vec{a}, \vec{b} \in \mathcal{D}$, $\vec{a} \neq \vec{b}$, and $\mathcal{D}$ 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 $\mathcal{D}_f$.*

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

$$|\mathcal{D}_f| \leq \sum_{(i<j)} k_i k_j,$$

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

**Algorithm 6.3** (ALG3: Reduction of Compound Variables )

1) Derive the set of difference vectors $\mathcal{D}_f$ of an $n$-variable function.
2) If $|\mathcal{D}_f| = 2^n - 1$, then stop, since reduction is impossible.
3) Obtain a non-zero vector $\vec{d} \in \mathcal{B}^n \backslash \mathcal{D}_f$ 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 methods, we reduced the compound variables for various benchmark functions [9]. Table 7.2 shows the results. The first four columns show the attributes 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 column ALG3 shows the number of compound variables obtained by Algorithm 6.3. The columns headed with $s$-MIN ($s = 3, 4, 5$) show the number of variables obtained by $s$-MIN when the transformation matrices are $A$, $C$, $E$, respectively. The columns headed with $s$-min ($s = 3, 4, 5$) show the number of variables obtained by $s$-MIN when the transformation matrices are $B$, $D$, $F$, respectively. $s$-MIN and $s$-min were applied to the results of ALG1.

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

ALG1 $>$ 2-MIN $>$ ALG2 $>$ 3-MIN $>$4-MIN $\simeq$ 5-MIN$>$ ALG3.

Also, $s$-MIN produced solutions with fewer variables than $s$-min.

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. ALG1 requires the shortest CPU time, while ALG3 requires the longest CPU time, when $k$ is large.

TABLE 7.1
Complexity of Reduction Algorithms

| Algorithm | Degree | CPU Time | Memory |
|---|---|---|---|
| ALG1 | 1 | $O(nk \log k)$ | $O(nk)$ |
| ALG2 | 2 | $O(n^2k \log k)$ | $O(n^2k)$ |
| ALG3 | 3+ | $O(nk^2)$ | $O(nk^2)$ |
| $s$-MIN | 3+ | $O(n^s k \log k)$ | $O(nk)$ |

## VIII. An Application as a Pre-Processor

Table 7.2 shows that ALG3 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 and 4-MIN to reduce the total CPU time for MNIST 28×28 function. This function represents a handwritten digits recognition circuit [2]. It has $n_0 = 784$ variables, $k = 59981$ registered vectors, and $m = 10$ classes. ALG1 yields $n_1 = 37$-variable solution in 39

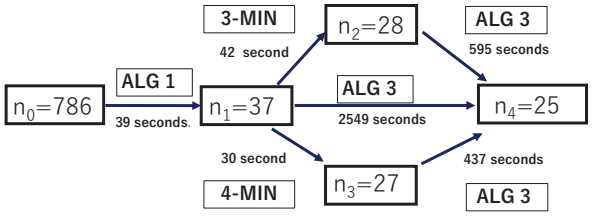


Fig. 7.1. Reduction of total CPU time for MNIST 28 × 28.

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

TABLE 7.2
RESULTS OF LINEAR TRANSFORMATIONS

| Function Data | | | | Number of Variables | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Existing Methods | | | | | New Methods | | | | |
| Name | $n$ | $m$ | $k$ | ALG1 | ALG2 | ALG3 | 2-MIN | 3-MIN | 3-min | 4-MIN | 4-min | 5-MIN | 5-min |
| 4350WORDS | 75 | 14 | 4,350 | 43 | 25 | 18 | 21 | 20 | 21 | 19 | 20 | 20 | 20 |
| CHESS3196 | 75 | 2 | 3,196 | 30 | 21 | 15 | 18 | 16 | 17 | 17 | 16 | 17 | 17 |
| CIFAR $32 \times 32$ | 1024 | 2 | 9,930 | 58 | 30 | 19 | 25 | 22 | 24 | 21 | 23 | 21 | 22 |
| COMPANIES | 30 | 9 | 3,700 | 21 | 19 | 18 | 20 | 19 | 20 | 19 | 19 | 19 | 20 |
| CONNECT-4 | 126 | 3 | 67,557 | 61 | 38 | 22 | 34 | 28 | 32 | 25 | 29 | 25 | 27 |
| LETTER-RECOG. | 256 | 26 | 20,000 | 51 | 32 | 21 | 36 | 28 | 35 | 25 | 32 | 24 | 29 |
| MNIST $14 \times 14$ | 196 | 10 | 58,191 | 45 | 29 | 25 | 32 | 28 | 29 | 27 | 29 | 27 | 28 |
| MNIST $28 \times 28$ | 784 | 10 | 59,981 | 37 | 29 | 25 | 30 | 28 | 30 | 27 | 28 | 27 | 29 |
| MNIST 2CLASS | 784 | 2 | 59,984 | 35 | 25 | 24 | 31 | 27 | 30 | 26 | 30 | 26 | 29 |
| MNIST $8 \times 8$ | 64 | 10 | 3,686 | 23 | 19 | 17 | 20 | 19 | 20 | 19 | 20 | 19 | 20 |
| NURSERY | 27 | 5 | 12,960 | 17 | 15 | 14 | 17 | 14 | 17 | 15 | 17 | 15 | 17 |
| POKER | 85 | 10 | 25,010 | 61 | 37 | 21 | 43 | 23 | 37 | 24 | 29 | 24 | 28 |
| RANDOM4000 | 30 | 4 | 4,000 | 21 | 20 | 18 | 20 | 19 | 19 | 19 | 19 | 19 | 22 |
| SPAM MAIL FILTER | 128 | 2 | 20,000 | 23 | 22 | 22 | 23 | 23 | 22 | 23 | 23 | 23 | 23 |
| SPLICE | 240 | 3 | 3,174 | 18 | 17 | 15 | 18 | 17 | 18 | 18 | 18 | 17 | 18 |
| Total | | | | 544 | 379 | 294 | 388 | 331 | 371 | 324 | 352 | 323 | 349 |

seconds. While, ALG3 yields an $n_3 = 25$-variable solution in 2549 seconds, which is very time-consuming.

However, we can reduce total CPU time by using 3-MIN as shown in Fig. 7.1. First, we apply ALG1 to obtain an $n_1 = 37$-variable solution. Second, we apply 3-MIN to obtain an $n_2 = 28$ variable solution in 42 seconds. Third, we apply ALG3 to obtain an $n_3 = 25$-variable solution in 595 seconds.

When, we apply 4-MIN to the result of ALG1, we obtained an $n_4 = 27$ variable solution in 30 seconds. Then, we apply ALG3 to obtain an $n_3 = 25$-variable solution in 437 seconds.

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. When we apply ALG3 to the result of Algorithm ALG1, the number of distinct difference vectors is 1,226,244,862.

However, when we apply ALG3 to the result of 3-MIN, the number of distinct difference vectors is reduced to 258,383,446. Also, the number of variables is reduced from 37 to 28, and the number of the iterations in ALG3 is reduced from 12 to 3.

On the other hand, when we apply ALG3 to the result of 4-MIN, the number of distinct difference vectors is reduced to 134,086,094. Also the number of variables is reduced from 37 to 27, and the number of the iterations in ALG3 is reduced from 12 to 2. In these ways, total CPU time to obtain a minimal solution can be reduced.

We used a computer with an INTEL Core i9, 10900, 2.8GHz CPU, and 64 GB main memory, on Windows 11.

## IX. CONCLUSION

This paper presented $s$-MIN, minimization algorithms.

1) They are faster and require smaller memory than ALG3.
2) Experimental results using various benchmark functions show that the average number of variables have tendency that

$$\text{2-MIN} > \text{3-MIN} > \text{4-MIN} \simeq \text{5-MIN}.$$

3) In addition to $s$-MIN, we tested $s$-min that use different linear transformations, but they produced inferior solutions to $s$-MIN.
4) These algorithms are suitable for 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. Reviewers comments improved the presentation of the paper.

## 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, "Linear transformations for variable reduction," *Reed-Muller 2011 Workshop*, Tuusula, Finland, May 25-26, 2011.
[5] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.
[6] T. Sasao, "Linear decomposition of index generation functions," *17th Asia and South Pacific Design Automation Conference* (ASPDAC-2012), 2012, Sydney, Australia, pp. 781-788.
[7] 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.
[8] T. Sasao, *Index Generation Functions*, Springer, Oct. 2019.
[9] T. Sasao, *Classification Functions for Machine Learning and Data Mining*, Springer Nature, Aug. 2023.
[10] T. Sasao, "Iterative linear transformation to reduce compound variables," *Workshop on Synthesis And System Integration of Mixed Information technologies*, (SASIMI-2024), March 11, Taipei, Taiwan.