IBIMA Publishing
Journal of Software & Systems Development
https://ibimapublishing.com/articles/JSSD/2021/590033/
Vol. 2021 (2021), Article ID 590033, 14 pages, ISSEN: 2166-0824
DOI: 10.5171/2021.590033

Research Article

Heuristic Minimization of Symmetric Index Generation Functions utilizing their Properties

Tomasz MAZURKIEWICZ

Faculty of Cybernetics, Military University of Technology, Warsaw, Poland,
tomasz.mazurkiewicz@wat.edu.pl, ORCID: 0000-0001-7305-2379

Received date: 9 November 2020; Accepted date: 15 January 2021; published date: 22 February 2021

Academic Editor: Iulian Furdu

Abstract

This paper focuses on the minimization of index generation functions, which are useful in telecommunication and cybersecurity. In particular, we target a specific class of those functions, i.e., symmetric index generation functions. The application of logic synthesis methods to such functions can often lead to a representation that uses fewer variables. We analyze the properties of those functions and how they can be used to minimize the number of variables. What is more, we investigate the influence on accelerating other minimization techniques, such as linear decomposition. Presented results show that by taking advantage of the described properties, the computations using the heuristic algorithm are significantly faster.

Keywords: Index generation functions, Logic synthesis, Linear decomposition, Symmetric functions, Variable reduction

Introduction

Memory-based logic synthesis of index generation functions (IGF) gained a lot of interest due to the applications of those functions (Sasao (2011, 2017, 2020)) in realizing pattern matching circuits. In the literature, there are many examples of using those functions in telecommunication, cybersecurity, and the Internet of Things for devices such as virtual routers and malicious data detection systems. What is more, the concept of a hardware implementation of index generation functions has been reported by Sasao (2020) as the key device in the network hardware.

Index generation function maps unique integer value $\{1,2,\dots,K\}$ to elements of a set

Cite this Article as: Tomasz MAZURKIEWICZ (2021)," Heuristic Minimization of Symmetric Index Generation Functions utilizing their Properties", Journal of Software & Systems Development, Vol. 2021 (2021), Article ID 590033, DOI: 10.5171/2021.590033

that consists of K different binary vectors, whose length equals N. Other assignments are left unspecified and function returns zero. The important property of those functions is that they are not fully defined, i.e., $K \ll 2^N$. Thus, they can be efficiently minimized and implemented using fewer variables than N.

In the literature, techniques such as variable reduction (Borowik and Łuba (2014), Sasao (2020)) and functional decomposition (Mazurkiewicz (2020a, 2020b), Nagayama et al. (2020)) are used very often to minimize memory usage. However, linear decomposition (using XOR gates) has been identified as being very efficient (Łuba et al. (2016), Mazurkiewicz and Łuba (2019), Sasao (2017, 2020)), i.e., for *M-out-of-N* coders. Interestingly, logic synthesis methods can be easily transferred and adapted to the field of knowledge discovery and data mining (Borowik (2014), Borowik and Łuba (2014)), i.e., to minimize the number of attributes and remove redundant decision rules.

Using the linear decomposition, an index generation function is implemented as a composition of a linear transformation L (using EXOR gates) and a general function G that is implemented using memory (RAM/ROM). Linear transformation, i.e., a composition of several linear functions, reduces the number of variables from N to P. Thus, $2^P Q$-bit memory is required to implement function G.

In this paper, we analyze how the properties of symmetric index generation functions, i.e., their structure, influence the process of their minimization. In particular, we focus on the application of the existing heuristic linear decomposition algorithm by Mazurkiewicz and Łuba (2019). Symmetric index generation functions are a special class of index generation functions. We prove that existing heuristic methods can be easily applied to minimize this class of functions. According to our best knowledge, other authors (Nagayama et al. (2020)) addressed only optimum linear decomposition algorithms.

In this paper, we also prove that properties of symmetric index generation functions can be used to simplify the computations, i.e., the number of variables N can be efficiently reduced. The approach proposed in this paper was presented in Fig. 1. Since $N \geq N' \geq P$, typically less memory needs to be used to implement the input function.

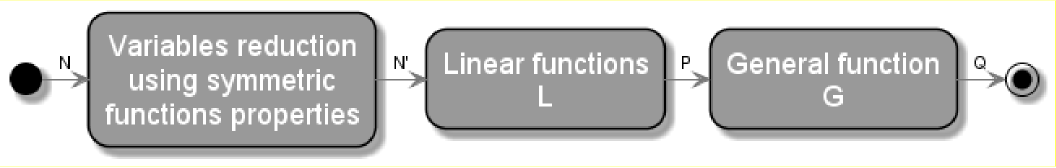


Fig. 1: Proposed approach

The rest of this paper is organized as follows: the second section defines symmetric index generation functions and compound degree. The linear decomposition algorithm using discernibility sets is described in the next section. In Section 4, we analyze the properties of symmetric index generation functions and their influence on the number of variables, i.e., we analyze S_1^N functions. We also extend our previous work by Mazurkiewicz (2020c) and present the properties of S_2^N functions. The results obtained using experimental software are presented in Section 5. The efficiency of the heuristic linear decomposition algorithm using the variable reduction method described in this paper is investigated in that section. The last section concludes the paper.

Preliminaries

An **index generation function** represents the following mapping:

$$F\colon D^N \rightarrow \{1,2,\dots,K\} \tag{1}$$

where D^N is a set of different binary vectors, called registered vectors, i.e., $D^N \subseteq \{0,1\}^N$. The important property of this function is that $|D^N| = K \ll 2^N$. Function F assigns the corresponding index (unique integer value from 1 to K) for every vector $v \in D^N$.

A **characteristic function** χ of an index generation function is

$$\chi : \{0,1\}^N \rightarrow \{0,1\} \tag{2}$$

where

$$\chi(v) = \begin{cases} 1 & \Leftrightarrow \quad v \in D^N \\ 0 & \Leftrightarrow \quad otherwise \end{cases} \tag{3}$$

In this paper, we focus on **symmetric index generation functions**, i.e., index generation functions whose characteristic function is symmetric.

M-out-of-N coders are often used as standard benchmark functions of linear decomposition algorithms. Typically, $M \in \{1,2,3,4\}$ and $N \in \{16,20\}$. Those coders consist of $K = \binom{N}{M}$ binary vectors, whose length is N and Hamming weight is M. Such functions represent symmetric index generation functions (Sasao (2020)) and are denoted S_M^N. An example of such a function was presented in Table 1.

Table 1: Symmetric function S_1^6.

x_1	x_2	x_3	x_4	x_5	x_6	$F(X)$
1	0	0	0	0	0	1
0	1	0	0	0	0	2
0	0	1	0	0	0	3
0	0	0	1	0	0	4
0	0	0	0	1	0	5
0	0	0	0	0	1	6

DOI: 10.5171/2021.590033

What is important, the classical variable reduction methods fail for S_M^N functions and reduce only one variable. Therefore, linear decomposition algorithms are mostly used to minimize those functions. In that case, an input function *F(X)* is realized using a general function *G(Y)* and *P* linear functions. Consider a linear function $y = \bigoplus_{i=1}^{N} x_i c_i$, where $x_i \in X$, $c_i \in \{0,1\}$. The compound degree of such function equals $\delta = \sum_{i=1}^{N} c_i$ (each c_i is viewed as an integer). Therefore, the compound degree of the function *G(Y)* is equal to $\max_{i=1,2,\dots,P} \delta$.

Linear Decomposition

To find a linear decomposition of an index generation function, a discernibility set might be used. In this section, we shortly present an iterative algorithm using such sets. For a detailed description please refer to the original paper by Mazurkiewicz and Łuba (2019). The efficiency of this algorithm for general index generation functions was already proven.

A discernibility set, denoted as C_{pq} is defined as follows:

$$C_{pq} = \{x \in X, p \neq q : x(p) \neq x(q)\} \tag{4}$$

p and q are indexes of registered vectors. Each vector $v \in D^N$ has index p iff *F(v) = p*.

The collection of all C_{pq} sets, i.e., for $p, q \in \{1,2,\dots,K\}$ and $p < q$, will be denoted as RC_{pq} and its complement as $COM(RC_{pq})$. Additionally, the complement limited to sets with cardinality r will be denoted as $COM(RC_{pq}^{r}), r \leq N$.

To find the decomposition of a function using several linear functions, the simple test can be used (Łuba et al. (2016), Mazurkiewicz and Łuba (2019)). Function $x_i \oplus x_j$ $(x_i, x_j \in X)$ is a decomposition function of F iff $\{x_i, x_j\} \notin RC_{pq}$. In that case, $\{x_i, x_j\} \in COM(RC_{pq})$. This simple test can be generalized. For example, pair of two linear functions: $x_i \oplus x_j$ and $x_j \oplus x_k$ $(x_k \in X)$ are decomposition functions of F iff $\{x_i, x_j, x_k\} \in COM(RC_{pq})$.

The discernibility matrix can be used to represent RC_{pq}. To improve the time efficiency of the algorithm, the repeating values are removed from this matrix. To illustrate the idea of discernibility sets, all C_{pq} sets generated for the S_1^6 function, i.e., RC_{pq}, were presented in Table 2. Additionally, the discernibility matrix was presented in Table 3. In each row, $x_i = 1$ iff $x_i \in C_{pq}$. Notice that each row can be generated using EXOR operation on p-th and q-th vectors from the truth table.

Table 2: RC_{pq} for symmetric function S_1^6.

p, q	C_{pq}	p, q	C_{pq}	p, q	C_{pq}
1,2	$\{x_1, x_2\}$	2,3	$\{x_2, x_3\}$	3,5	$\{x_3, x_5\}$
1,3	$\{x_1, x_3\}$	2,4	$\{x_2, x_4\}$	3,6	$\{x_3, x_6\}$
1,4	$\{x_1, x_4\}$	2,5	$\{x_2, x_5\}$	4,5	$\{x_4, x_5\}$

1,5	$\{x_1, x_5\}$	2,6	$\{x_2, x_6\}$	4,6	$\{x_4, x_6\}$
1,6	$\{x_1, x_6\}$	3,4	$\{x_3, x_4\}$	5,6	$\{x_5, x_6\}$

Table 3: The discernibility matrix for symmetric function S_1^6.

x_1	x_2	x_3	x_4	x_5	x_6	p, q
1	1	0	0	0	0	1,2
1	0	1	0	0	0	1,3
1	0	0	1	0	0	1,4
1	0	0	0	1	0	1,5
1	0	0	0	0	1	1,6
0	1	1	0	0	0	2,3
0	1	0	1	0	0	2,4
0	1	0	0	1	0	2,5
0	1	0	0	0	1	2,6
0	0	1	1	0	0	3,4
0	0	1	0	1	0	3,5
0	0	1	0	0	1	3,6
0	0	0	1	1	0	4,5
0	0	0	1	0	1	4,6
0	0	0	0	1	1	5,6

To find linear functions that decompose an index generation function, an iterative approach was proposed (Mazurkiewicz and Łuba (2019)). The algorithm was presented as Algorithm 1. Some speed-up techniques were described in the original paper.

In this paper, we assume that the input function is not preprocessed using the argument reduction procedure (step 1 in Algorithm 1). For symmetric functions S_M^N, this procedure reduces only one variable and leads to a single row of all zeroes, which heavily affects the properties of analyzed functions. Furthermore, in our experiments, this step is replaced with the procedure described in Section 4.

The mentioned algorithm works as follows. Firstly, discernibility sets are calculated and used to form a discernibility matrix (step 2). The repeating values are removed from the matrix (step 3). Next, the algorithm searches for a decomposition function of F. As long as $COM(RC_{pq}) \neq \emptyset$, further decomposition can be found. Thus, the algorithm iteratively finds a decomposition function and regenerates the discernibility matrix. The discernibility matrix is regenerated based on a linear transformation g that was found in each iteration (steps 5 and 6). Duplicates are removed again to increase the efficiency of the procedure of searching for a decomposition function.

Algorithm 1. Linear decomposition algorithm by Mazurkiewicz and Łuba (2019)

```
    **Input:** function $F$
    **Output:** minimized function
1:  $F \leftarrow variable\_reduction(F)$
2:  $DM \leftarrow generate\_disernibility\_matrix(F)$
3:  $DM \leftarrow remove\_duplicates(DM)$
4:  $\boldsymbol{while}\ COM(RC_{pq})\ \boldsymbol{is\ not}\ \emptyset\ \boldsymbol{do}$
5:      $g \leftarrow find\_decomposition\_function(DM)$
6:      $DM \leftarrow modify\_discernibility\_matrix(DM, g)$
7:      $F \leftarrow modify\_function(F, g)$
8:  $\boldsymbol{end\ while}$
9:  $\boldsymbol{return}\ F$
```

The key element of the algorithm is the proper selection of a decomposition function (step 5). The most time-efficient approach, called First-Fit or simply FF, chooses the first (in the lexicographic order) decomposition that was found. Unfortunately, the solution provided by this method is not always optimal, i.e., for *M-out-of-N* coders.

Another approach called MinR was proposed to address this issue. It uses the number R of distinct row vectors in a truth table of a function limited to a subset of input variables $X \backslash d$, where d denotes variables used in a linear function and $|d| = r$. For example, if $y = x_1 \oplus x_2$, then $d = \{x_1, x_2\}$ and $X \backslash d = \{x_3, x_4, \dots, x_N\}$. The function that provides the minimum value of R is chosen as a decomposition function. If two or more subsets have the same value of R, then the first function found is chosen. In that case, we proceed similarly to the FF method.

Due to the additional calculations in a decomposition selection procedure, the MinR approach is more time-consuming than the FF approach. However, the results presented in the literature (Mazurkiewicz and Łuba (2019)) prove that it provides better results for *M-out-of-N* coders. On average, both approaches lead to similar results in terms of the solution quality. Therefore, the FF approach is much more useful for typical index generation functions due to the time-efficiency.

What is important, in both approaches the algorithm searches in each iteration for a decomposition function with a compound degree as small as possible. Thus, firstly $COM(RC_{pq}^2)$ is analyzed, then $COM(RC_{pq}^3)$, and so on as long as $r \leq N$ and $COM(\mathrm{RC_{pq}}) \neq \emptyset$.

Properties of Symmetric Functions

In this section, we analyze the properties of symmetric functions. We determine how they can be used for reducing the number of variables. In particular, they are used to modify the truth table of a function in the first step of Algorithm 1. In the first and second subsections, we focus on symmetric index generation functions with $M = 1$, i.e., S_1^N. The last subsection extends previous work by Mazurkiewicz (2020c) and deals with S_2^N functions.

As mentioned earlier, the classical variable reduction methods fail for S_M^N functions. Thus, the approach described in this section is really useful to speed-up other minimization techniques by reducing the number of input variables. The correctness of the theoretical consideration presented in this section was practically confirmed using experimental software (written in Python) and the functions that are presented in the next section.

Basic Minimization

Let $M = 1$ and $N \geq 3$. In that case, $K = N$. In a symmetric index generation function S_1^N variable $x_i = 1$ $(x_i \in X)$ in each registered vector $v \in D^N$ iff $i = F(v)$. Recall that Table 1 illustrates this property. In that case, any C_{pq} set contains only two variables: x_p and x_q since registered vectors with indexes p and q will always differ on those two variables. Therefore, the generation of RC_{pq} leads to a discernibility matrix that contains all possible vectors, whose length is N, and Hamming weight is two. Any variable $x_i = 1$ iff $i = p \vee i = q$. Thus, $COM\left(RC_{pq}^2\right) = \emptyset$ and no function with compound degree two can be used as a decomposition function.

On the other hand, RC_{pq} does not contain any set with a cardinality bigger than two. Therefore, since $\{x_1, x_2, x_3\} \in COM\left(RC_{pq}\right)$ the algorithm chooses a pair of functions: $y_1 = x_1 \oplus x_2$ and $y_2 = x_2 \oplus x_3$ as decomposition functions in the FF approach. Recall that the MinR approach requires additional computations to select a function. However, the discernibility matrix of the analyzed symmetric index generation function can be also treated as a symmetric function. Thus, the value of R will be the same for all possible subsets d. This leads to the conclusion that the same pair of functions is chosen also in the MinR approach.

Notice that a pair of functions will be used as decomposition functions for any symmetric index generation function S_1^N. What is important is that the pair is known in advance without any computations whatsoever. The number of variables is reduced from N to $N' = N - 1$.

Consider now the function after first iteration of the algorithm if $N \geq 6$, i.e., $F(X) = G(x_4, x_5, \ldots, x_N, y_1, y_2)$. Notice that first and third vectors are vectors whose Hamming weight equals one. On the other hand, the second vector has 1 on both y_1 and y_2 variables. Thus, the truth table of a function still contains all possible vectors with Hamming weight one. The length of those vectors equals $N' = N - 1$. Therefore, $COM\left(RC_{pq}^2\right) = \emptyset$ one more time. Decomposed function S_1^6 and all calculated C_{pq} sets with cardinality two were presented in Tables 4 and 5 accordingly to illustrate described theoretical considerations.

Table 4: Decomposition of symmetric function S_1^6.

(a) Function after first iteration.

x_4	x_5	x_6	y_1	y_2	$G(X')$
0	0	0	1	0	1
0	0	0	1	1	2
0	0	0	0	1	3
1	0	0	0	0	4
0	1	0	0	0	5
0	0	1	0	0	6

(b) Function after second iteration.

y_1	y_2	y_3	y_4	$G(X'')$
1	0	0	0	1
1	1	0	0	2
0	1	0	0	3
0	0	1	0	4
0	0	1	1	5
0	0	0	1	6

Table 5: C_{pq}^2 sets in the second iteration.

p,q	C_{pq}^2	p,q	C_{pq}^2
1,3	$\{y_1, y_2\}$	3,5	$\{y_2, x_5\}$
1,4	$\{y_1, x_4\}$	3,6	$\{y_2, x_6\}$
1,5	$\{y_1, x_5\}$	4,5	$\{x_4, x_5\}$
1,6	$\{y_1, x_6\}$	4,6	$\{x_4, x_6\}$
3,4	$\{y_2, x_4\}$	5,6	$\{x_5, x_6\}$

Using the FF approach, we proceed similarly as previously. This leads to choosing a pair of functions $y_3 = x_4 \oplus x_5$ and $y_4 = x_5 \oplus x_6$ as decomposition functions. Notice that for the FF approach the described procedure will be used iteratively $\zeta = \left\lfloor \frac{N}{3} \right\rfloor$ times. Thus, the number of variables can be reduced to $N' = N - \zeta$ without any computations. In the next subsection, we investigate if some additional improvement can be achieved.

On the other hand, for the MinR approach, we have to proceed differently after finding the first pair of decomposition functions. Notice that if subset d contains only variables x_i, then all rows in a truth table are distinct. Thus, $R = K$. Recall that in the MinR approach, we look for a subset d that minimizes the value of R. If $y_1 \in d$, then the second and third rows are the same, i.e., all variables x_i are zeroes and $y_2 = 1$. Other rows are distinct. Thus, $R = K - 1$ and pair of functions: $x_4 \oplus x_5$ and $x_5 \oplus y_1$ is chosen as decomposition functions. In the end, the number of variables is reduced by two without any computations.

Further Minimization

From this point, we analyze only the FF approach. Let $N = 3 \cdot n$, $n \in \mathbb{N}$ and $F(X) = G(y_1, y_2, \dots, y_P)$. If $n = 2$, then $COM(RC_{pq}) \neq \emptyset$ and no further minimization is possible. If $n > 2$, then the vectors from the truth table of a function can be represented as a composition of the following two matrices:

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}_{3\times 2} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}_{3\times 2} \tag{5}$$

In the result, we get the following matrix:

$$M = \begin{bmatrix} A & B & \cdots & B \\ B & A & \cdots & B \\ \vdots & \vdots & \ddots & \vdots \\ B & B & \cdots & A \end{bmatrix}_{n\times(2\cdot n)} \tag{6}$$

Notice that matrix B contains only zeroes. Matrix A contains two rows with Hamming weight equal to one and one with Hamming weight equal to two. What is more, this matrix contains all possible vectors, whose length is two, and Hamming weight is one. In the matrix M, every row and every column contain only one matrix A. Thus, the matrix M contains all possible vectors, whose length is $2 \cdot n$, and Hamming weight is one. In that case, $COM\left(RC_{pq}^{2}\right) = \emptyset$ and no function with compound degree equal to two can be used as a decomposition function.

Since matrix A contains a row of all ones, it is not possible to find a pair of decomposition functions $y_i \oplus y_j$ and $y_j \oplus y_k$ such that $j - i = 1 \vee k - j = 1$. Thus, the first (in the lexicographic order) pair of decomposition functions that satisfies this limitation is chosen, i.e., $y_1 \oplus y_3$ and $y_3 \oplus y_5$. In that case, the number of variables was reduced to $N' = \frac{2}{3}N - 1$ without any computations.

On the other hand, if $N \neq 3 \cdot n$ $(N \geq 7)$, then we have $F(X) = G(x_N, y_1, y_2, \dots, y_{P-1})$ or $F(X) = G(x_{N-1}, x_N, y_1, y_2, \dots, y_{P-2})$. In that case, instead of the matrix M, we get one of the following matrices accordingly:

$$M_1 = \begin{bmatrix} 0 & M \\ 1 & 0 \end{bmatrix}_{(n+1)\times(2\cdot n+1)} \quad (7)$$

$$M_2 = \begin{bmatrix} 0 & 0 & M \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_{(n+2)\times(2\cdot n+2)} \quad (8)$$

Recall that the matrix M contains all possible rows with Hamming weight equal to one, whose length is $2 \cdot n$. Thus, both M_1 and M_2 matrices contain all possible $2 \cdot n + 1$ and $2 \cdot n$+2-bit length vectors accordingly. Therefore, again $COM\left(RC_{pq}^{2}\right) = \emptyset$.

Notice that the M_1 matrix contains a row such that only y_1 and y_2 are one. Thus, $\{x_N, y_1, y_2\} \in RC_{pq}$ and a pair of functions $x_N \oplus y_1$ and $y_1 \oplus y_2$ cannot be used as decomposition functions. On the other hand, $\{x_N, y_1, y_3\} \in COM\left(RC_{pq}\right)$ and one more variable is reduced without computations. Similarly, the M_2 matrix contains a row such that only y_1 is one. Thus, $\{x_{N-1}, x_N, y_1\} \in RC_{pq}$ and $\{x_{N-1}, x_N, y_2\} \in COM\left(RC_{pq}\right)$.

As a result, for both matrices, we obtain a reduction of variables by $\left\lfloor \frac{N}{3} \right\rfloor - 1$ without any computations whatsoever.

Several remarks on S_2^N functions

Let $M = 2$. In that case, a truth table of symmetric index generation function contains all $K = \binom{N}{2}$ binary vectors, whose length is N and Hamming weight is two. In this subsection, we consider S_2^4 function and present the analysis of the properties of the $\mathrm{S}_2^{\mathrm{N}}$ functions.

The truth table of the S_2^4 function is presented in Table 6. Notice that x_1 equals one for the first three vectors. In that case, the rest of the variables, i.e., x_2, x_3 and x_4, represent all possible vectors, whose length is $N - 1$ and Hamming weight is one. Thus, a discernibility matrix contains all possible vectors with $x_1 = 0$, whose Hamming weight is two, i.e., rows that represent C_{12}, C_{13} and C_{23} sets.

DOI: 10.5171/2021.590033

Table 6: Symmetric function S_2^4.

x_1	x_2	x_3	x_4	$F(X)$
1	1	0	0	1
1	0	1	0	2
1	0	0	1	3
0	1	1	0	4
0	1	0	1	5
0	0	1	1	6

It should be also noted that for other vectors x_1 equals zero and their Hamming weight is two. Thus, for $p = \{1,2,3\}$ and $q = \{4,5,6\}$, sets C_{pq} contain all possible vectors with $x_1 = 1$ and Hamming weight two. In that case, the generation of RC_{pq} leads to a discernibility matrix that contains all possible vectors, whose length is N and Hamming weight is two. It also contains rows of all ones, e.g., $C_{16} = \{x_1, x_2, x_3, x_4\}$. The obtained discernibility matrix (before removing duplicates) was presented in Table 7. Based on that, the algorithm chooses a pair of functions to decompose the input function, i.e., $y_1 = x_1 \oplus x_2$ and $y_2 = x_2 \oplus x_3$ $\left(\{x_1, x_2, x_3\} \in COM\left(RC_{pq}\right)\right)$.

Table 7: The discernibility matrix for symmetric function S_2^4.

x_1	x_2	x_3	x_4	p,q
0	1	1	0	1,2
0	1	0	1	1,3
1	0	1	0	1,4
1	0	0	1	1,5
1	1	1	1	1,6
0	0	1	1	2,3
1	1	0	0	2,4
1	1	1	1	2,5
1	0	0	1	2,6
1	1	1	1	3,4
1	1	0	0	3,5
1	0	1	0	3,6
0	0	1	1	4,5
0	1	0	1	4,6
0	1	1	0	5,6

For bigger values of N ($N \geq 6$), it should be noted that set of vectors for a subset of input variables, i.e., $\{x_4, x_5, \dots, x_N\}$, contains all possible vectors with Hamming weight equal to one and two, and vectors of all zeroes. Using considerations described earlier in this section, we know that $\{x_4, x_5, x_6\} \in COM(RC_{pq})$. Thus, $y_3 = x_4 \oplus x_5$ and $y_4 = x_5 \oplus x_6$. This leads to a conclusion that the approach for S_2^N functions is quite similar to that described in Section 4a of this paper. We can repeat this procedure ζ times in total. Therefore, the number of variables is reduced to $N' = N - \zeta$ without any computations.

Evaluation

To evaluate the efficiency of the described approach, we analyze some basic symmetric index generation functions. The same functions were used by other authors (Nagayama et al. (2020)) to analyze the optimum algorithm. Table 8 shows obtained results, i.e., the reduction of the number of variables (from N to N'), using the described properties of symmetric functions and the FF approach. In table 8a, *S. 4a* denotes results obtained using the approach described in Section 4a of this paper. Column *S. 4b* shows the results after applying further minimization. Notice that the number of variables was significantly reduced without any computations. The values of N, K and the number of variables obtained using the approach from Section 4c for S_2^N functions were presented in Table 8b. The linear decomposition algorithms can be applied to further reduce the number of variables. Since the time efficiency of all heuristic linear decomposition algorithms depends on the number of input variables, reducing it leads to better efficiency.

Table 8: Obtained results using the properties of symmetric functions and the FF approach

a) S_1^N functions

Function	*N*	*S. 4a*	*S. 4b*
S_1^{10}	10	7	6
S_1^{20}	20	14	13
S_1^{30}	30	20	19
S_1^{40}	40	27	26
S_1^{50}	50	34	33
S_1^{60}	60	40	39
S_1^{70}	70	47	46
S_1^{80}	80	54	53

b) S_2^N functions

Function	*N*	*K*	*S. 4c*
S_2^{10}	10	45	7
S_2^{15}	15	105	10
S_2^{20}	20	190	14
S_2^{25}	25	300	17
S_2^{30}	30	435	20
S_2^{35}	35	595	24

Table 9 shows the obtained results after minimizing symmetric functions using original heuristic algorithms by Mazurkiewicz and Łuba (2019). The column denoted *Opt* shows results obtained using the optimum method by Nagayama et al. (2020). However, this method searches only for solutions with a compound degree less or equal to five. Thus, it is possible to find a decomposition with fewer input variables than *Opt*. The column denoted κ presents the optimum theoretical solution calculated using the following formula:

$$\kappa = \lceil \log_2 K \rceil \qquad (9)$$

Table 9: Results obtained using heuristic algorithms

Function	***Opt***	***κ***	FF	MinR
S_1^{10}	4	4	4	4
S_1^{20}	7	5	6	5
S_1^{30}	10	5	6	5
S_1^{40}	13	6	7	6
S_1^{50}	17	6	7	6
S_1^{60}	20	6	7	6
S_1^{70}	23	7	8	7
S_1^{80}	27	7	8	7

Table 10 shows the compound degree for results obtained using FF and MinR methods. The most remarkable result to emerge from the data is that the application of heuristic methods leads to better results in terms of the number of input variables. Results obtained using the FF approach are close to κ, while the MinR method leads to results equal κ for all analyzed functions. However, the obtained compound degrees are generally high. It is worth mentioning that both methods, FF and MinR, were not designed to minimize the compound degree of a decomposed function.

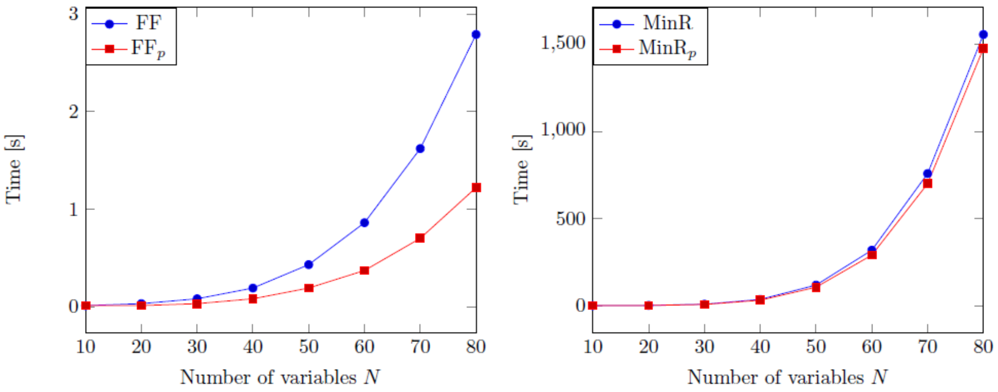
Using the approach described in this paper, we obtain worse result by one variable for the S_1^{60} function using the FF method. The main reason for this is that the argument reduction procedure is not applied. Thus, an input function is slightly different. For other analyzed symmetric functions, we get the same number of the variables P after linear decomposition.

Table 10: Compound degree.

Method	S_1^{10}	S_1^{20}	S_1^{30}	S_1^{40}	S_1^{50}	S_1^{60}	S_1^{70}	S_1^{80}
FF	4	9	15	22	29	29	38	36
MinR	5	10	16	20	32	32	26	40

Figure 2 shows computation time (in seconds) of heuristic decomposition of symmetric index generation functions S_1^N. The experiments were conducted using experimental software implemented in Python (using NumPy), and run on the following computer environment: CPU: Intel Xeon E5-2650v2 2.6Hz, memory: 64 GB, OS: Windows 7, interpreter: Python 3.8. Computation time using original algorithms was presented. Additionally, the computation time after reducing the number of variables using the approach proposed in this paper was presented to evaluate its efficiency. It was denoted FF_p and $MinR_p$ accordingly.

The FF method is much faster than the optimum method by Nagayama et al. (2020) implemented in the C language and leads to quite good results. For example, it finds the solution for the S_1^{80} function that is worse than the theoretical optimum by only one variable in 1.2 seconds. On the other hand, the time efficiency of the MinR method is much worse. Notice that the reduction of the number of variables leads to better time efficiency of both algorithms. Time FF$_p$ is up to 2.3 times shorter than the computation time of the original method. On the other hand, the MinR method has been accelerated only by 5%. Notice that the presented results prove that the MinR method is more time-consuming.



(a) The FF method.

(b) The MinR method.

Fig. 2: Computation time (in seconds) for S_1^N functions

Figure 3 shows the computation time (in seconds) of minimization of several S_2^N functions. The obtained results are similar to those for S_1^N functions. A significant reduction in computation time was achieved using the approach described in this paper. Notice that computation times are longer compared to those presented in Figure 2a. This is because the number of vectors K is much larger for S_2^N functions. For example, the truth table of the S_1^{35} function has $K = 35$ vectors. On the other hand, if $M = 2$, then $K = 595$ (see Table 8b).

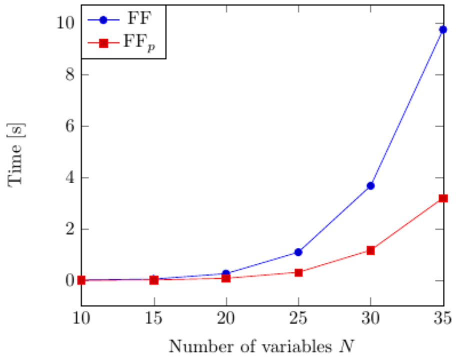


Fig. 3: Computation time (in seconds) for S_2^N functions

Conclusion

In this paper, the properties of symmetric index generation functions were analyzed. Additionally, we proved that those properties can be used to efficiently reduce the number of variables. This leads to a lower computation time of the heuristic linear decomposition algorithms. In particular, the FF method was accelerated by up to 2.3 times for analyzed S_1^N functions. Significant time improvement was also achieved for S_2^N functions. What is more, we proved that heuristic algorithms can provide better results (in a shorter time) than the optimum algorithm in terms of the number of variables P. However, the compound degrees are much higher.

In this paper, we focused on symmetric functions with $M \leq 2$. Therefore, our future work includes analysis of properties of functions with $M \geq 3$ and their influence on heuristic linear decomposition algorithms.

References

- Borowik, G. (2014), 'Data Mining Approach for Decision and Classification Systems Using Logic Synthesis Algorithms,' *Advanced Methods and Applications in Computational Intelligence. Topics in Intelligent Engineering and Informatics*, vol. 6, pp. 3-23.
- Borowik, G. and Łuba, T. (2014), 'Fast Algorithm of Attribute Reduction Based on the Complementation of Boolean Function,' ch. 2, pp. 25-41, Springer International Publishing.
- Łuba, T., Borowik, G. and Jankowski, C. (2016), 'Gate based decomposition of index generation functions,' *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments, Proc. of SPIE,* vol. 10031.
- Mazurkiewicz, T. (2020), 'Non-disjoint functional decomposition of index generation functions,' Proceedings of the 50th IEEE International Symposium on Multiple-Valued Logic (ISMVL), Miyazaki, Japan, pp. 137-142.
- Mazurkiewicz, T. (2020), 'Application of Graph Theory Algorithms in Non-disjoint Functional Decomposition of Specific Boolean Functions,' *Journal of Telecommunications and Information Technology*, pp. 67-74, 3/2020, https://www.il-pib.pl/czasopisma/JTIT/2020/3/67.pdf
- Mazurkiewicz, T. (2020), 'On heuristic linear decomposition of symmetric index generation functions,' Proceedings of the 36th International Business Information Management Association Conference (IBIMA), Granada, Spain, pp. 11011-11018.
- Mazurkiewicz, T. and Łuba, T. (2019), 'Linear and non-linear decomposition of index generation functions,' Proceedings of the 26th International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES), Rzeszów, Poland, pp. 246-251.
- Nagayama, S., Sasao, T. and Butler, J.T. (2020), 'On Optimum Linear Decomposition of Symmetric Index Generation Functions,' Proceedings of the IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL), Miyazaki, Japan, pp. 130-136.
- Sasao, T. (2011), Memory-Based Logic Synthesis, 1st ed., New York: Springer-Verlag.
- Sasao, T. (2017), 'Index generation functions: Minimization methods,' Proceedings of the IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL), Novi Sad, Serbia, pp. 197-206.
- Sasao, T. (2020), Index generation functions, Synthesis lectures on digital circuits and systems, San Rafael, CA: Morgan & Claypool Publishers.